The Initons Catalytic Reflection Between Humanoid DNA and Nero Cell

Yaoguang Luo, Rongwu Luo

Liuyang DETA Software Development Limited Company
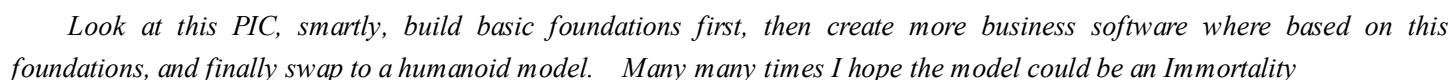
Changsha Lehao Fu Chan Hospital

Keywords

VPCS, AOPM, IDUC, Nero, Artificial, Decoder, Medical, Paralling, Computing, Humanoid, ETL, Parser, Data Mining

Outlook: VPCS architecture is not the end. Absolutely, At least at this paper, I will make an implementation in five section: DETA humanoid cognition, DETA Medical Business backend logic, DETA Catalytic computing, DETA Finding initions, DETA DNA decoding. Above all I also will spend more and more words in my DETA DNA Law of IDUC. And its applications in the real world. For more information and sources please find wechat ID: sweet00048, ok next step as below.

I DETA humanoid cognition

Since the inition of the DETA OSS, there has a lot of questions where based on the humanoid DNA catalytic computing, I have been working on this domain for a long time. Absolutely also, I have got a lot of flashing points here, for example AI, still remember the first time I touch the cognition this verbal at CLU Dr Renhat's class about cogs PU computing and cognition quality Sonar test where in Folsom Intel, I know that cog-work is a trending task in my life.

Eventhough I need do a lot of basic foundations at AI, but I insist to now, from the code where between normalization and duplication. I have been thinking how to make a contraction and distinction AI logic with human and humanoid. Ok , PLAN A starts as below. Code a problems solution software like a way of Yaoguang Luo 's cog-style life.

Look at this PIC, smartly, build basic foundations first, then create more business software where based on this foundations, and finally swap to a humanoid model.   Many many times I hope the model could be an Immortality

I I DETA humanoid cognition history

In the past, knowledge of the world could be parsed by five sections, the world's cognitive way, philosophers and scientists liked to describe objects with five senses (touch, taste, hearing, smell and vision). Species sense can make creatures in order to adapt to the environment well, understand the environment, and think about ways to protect themselves in dangerous environments. These methods are implemented in a variety of ways, and their execution logic is also varied, but the causes and results are clear about a basic point. Better adapt to the environment. During the process of designing the YangLiaoJing, the author has well integrated the bionic technologies of voice, text, association and visual media, and has been optimizing them to gradually form a comprehensive intelligent YangLiaoJing system.

I II DETA humanoid cognition development

In order to better adapt to the environment, human beings began to create characters, invent tools and improve their cognitive ability, from the ignorance of slaves to the open and compatible world, from the Iron Age to the current nano-chip technology. In the river history of 5000 years, human beings seem to have evolved aimlessly. Through these phenomena, the essence can be easily discovered. Better adapt to and transform the environment. The best arguments to improve the cognitive ability of environmental things are the research and development of basic science and technology and systematic induction, Marconi's wireless telegraph, Zu Chongzhi's pi, Darwin's origin of species, code of Hammurabi, and countless outstanding scientists, thinkers, inventors and philosophers in history. These people seem to be shining stars in the night sky. Gradually,

Intelligent creatures begin to have enough ability to look up into space and explore the mysteries of the universe, and this enough ability is the basic ability to improve the cognition of things, which is very important. The Huaruiji system of DETA Company draws lessons from the systematic induction method of human instinct and dialectically treats medical diseases with medical textbooks as the cognitive basis, which is in line with the embodiment of scientific development.

I III DETA humanoid cognition application

There are many applications of humanoid cognition in social science, and there are many excellent arguments here, such as auxiliary auditory system, big data reasoning system, weather forecasting system and criminal investigation database system, which undoubtedly proves that cognitive model has greatly improved the adaptability of human beings to the environment. To the ability to gradually transform the environment from part to whole. There are many outstanding arguments here, from the ancient Dayu flood control, to the renovation of Emperor Yangdi's Canal, to the renovation of General Secretary Xi Jinping's' green water and green mountains', from artificial rainfall to artificial islands. The argument is very clear, and the cognitive ability of things comes from the accumulation of basic science and technology. These basic technologies gradually form a system. On top of it are a wide range of scientific and technological commodity applications, such as hybrid rice of Yuan Longping team to improve material environment, comprehensive treatment scheme of SARS-CoV-2 of Zhong Nanshan team to improve healthy environment, giant hydropower station of Three Gorges of Yangtze River to improve geographical environment, etc. There are too many. The evolution of human wisdom gradually forms a clear route, and improving basic science and technology and cognitive ability complement each other. These abilities all come from thinking about things. Then form a solution, and finally implement it. This process is summarized as the process of analysis, operation, processing and management. The life cycle of software engineering is well explained here. Analysis A, Operation O, Processing P, Management M, use simple words to describe that even if unknown data are collected and analyzed, and then things are operated, the solutions to various difficulties encountered in the operation process are implemented. Finally, maintain and manage these implementation experiences. The back-end computing mode and system life cycle of DETA have gradually condensed from the earliest collection, analysis, operation, sorting, coding, running, debugging and maintenance to the module modes of Analysis A, Operation O, Processing P and Management M, such as DETA word segmentation, DETA DNN mind reading, etc. Now ETL of DETA is ready to go in this direction. The author designed a paper last year to describe the application mechanism of AOPM as follows:

AOPM Open Source System On SDLC Theory

Mr. Yaoguang. Luo

Liu Yang DETA Software Development Limited Company, Hunan, China,

313699483@qq.com

Outline: Mr. Xuesen. Qian once said: Science Is A Titan System, as an open source software conception. this topic implements a software interaction theory of SDLC for Analysis, Operation, Process and Management— AOPM. Also, this is a tiny paper where easy to show more idyllic landscapes of using DETA open source projects. Not only for web system, also for mobile and desktop platform. The final goal are makes complex project to simple. Ok let go and the next steps.

Keywords: SDLC, AOPM, VPCS, WEB, Concurrent, Open Source, Interaction, Management, Automation

Introductions

Recently my colleagues take more care on the SDLC evolution of open source software engineering, for each project they undertake on where it cost a lot of times, that's for my job, continuing found out a high effect, simple and clear theory of SDLC what be my main task now. after imagination and logic recursion, the key is an optimization of ordinary SDLC such as water

fall. First time for makes an introduction to waterfall of SDLC? The author's explanation likes sequence linked list of component nodes. With DETA projects here contains four aspects at Figure1-1. And my explanation of open source as belows.

*Topic: Ten Definition of The Open Source, OSS Book Reading Note/*这是我在路德大学研 *3* 写的读后感作业

In this paper, through a premise: the contrast between the copyright and the contract. the author talks a comprehensive introduction to the definition of the open source code. The role of the open source licenses, which is to allow the work permit under the non-exclusive business. Not only does it mean that the source code was visited by the public user, and also meets another 10 conditions as follows. The first point: the open source software allows the free reusable distribution. The license must not restrict that any party sell or give away the software. At the same time, it can't get the sold fees and other fees for this software. The second point: the program must include the full of source code. The license does not allow that getting the source code from any specific forms of the production. The license assures that no one can intentionally to confuse the source code. At the same time, the users have the right to access to the source code under this license. The third point: which talks about the rights of the derivative work. The license must allow the work-modification and the new-work -derivation . those new's are published under the same license. The fourth point: the integrity of the source code. Licenses and the integrity of permits,which may limit the distribution of the form of the modified source code. The fifth point: license does not discriminate against any specific groups and individuals. The sixth point: license does not limit the use way of any particular field scheme. At the same time, the license can't limit the use way's flexibility and reliability. The seventh point: the distribution of the license. Distribution solutions do not need additional license.The eighth point: the license must not specific to the product. The redistribution of the software does not dependent on the program. The ninth point: license may not restrict other software. This license may not restrict the publish of the software. The distribute software will be built by using open source. The end point: license rights is neutral. So, it effective limits that the freedom of the code transmission. In other words, it provides the preventive measures.
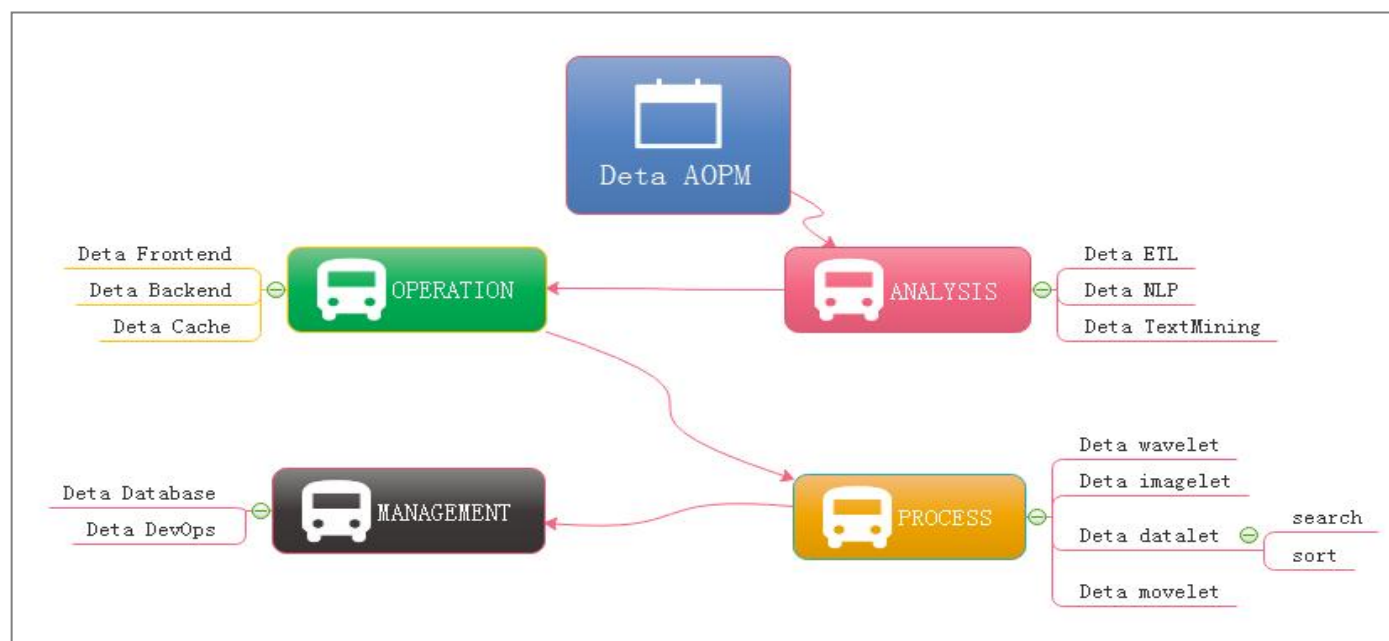

Figure 1-1 AOPM Applications with SDLC

Evolutions

Last year I was asked by so many engineers, almost the same question: how have you done so many projects during the year of 2018-19? My answer is absolutely: connection. Always, with connection, I got lots of fantasy inspirations on the projects where I undertook. My projects all are lower basic technical factors, with connections, what support me the necessary energy for continuing development on my projects. What means connection? Be an internal union bridge between my projects. For example DETA NLP and DETA ETL, they both have the same attributes such as AI, Analysis and Data etc, with this connections, my tasks became more dynamically. Everytime before I made a decision of priority levels of my projects, I

thought the connection first, DETA projects totally can be separated into three dimensions. Frontend Backend and Storage, as the Figure 1-2, the connection between DETA projects is WEB AI, now is a Bazaar requirement, but we will easy to make estimation of it's future, toward to Cathedral.

Cathedral and the Bazaar, this article has a profound implication, the author is a computer scientist with extensive experience. We can say that he is one of the early code and program contributors in the UNIX system. This article describes the Linux development with the revolutionary road, as the process from the bazaar to the cathedral. First, the author tells the contrast between UNIX and Linux: now UNIX is still popular around the world. Its rigorous structure and contribution to science, let it is proud of the same dignity as a church. Linux looks like a noisy bazaar, the code work in various countries around the world, to solve their own problems and arguing in the forums and communities. Like a bazaar. Then, author points an internal factors to get an in-depth discussion: UNIX reason why it has the church's authority, because its development has always been tailor-made by the world's most senior and most eminent researchers and software scientists. Although the discussion, because of the nature of the project-oriented, so that UNIX has been applied still to today. Even of the unreasonable original design, through decades of use, engineers have become accustomed to this experience now, there fore, we are called transcendental. which makes UNIX feels like a cathedral. The birth of the Linux was different, survival in an all-spittle environment. Every update, are implemented in controversial circumstances. The crowd here, are huge number of scientists, or writers,or code workers or merchants, their common ideal is that make Linux development meets the needs of all groups. Similar a huge bazaar. The author commenced a leno-vo, a conclusion that Linux will eventually beat UNIX, UNIX gets the range of fresh blood is less than the Linux's, also the number of the UNIX team members is less than the Linux's. UNIX customers and employees are aging. But Linux development more in line with the user of the needs. Its own development is to establish a relationship on this demand and requirement. Linux is young now. Summary, UNIX and Linux development option is the two kinds of very different road. These processes and methods to determine the fate of the two kinds of software development. Of more optimistic about Linux because it is better adapted to the environment.

At figure 1-2, DETA open source main based on AI domain, it already formed as an ecology system, go ahead to the application, thanks.
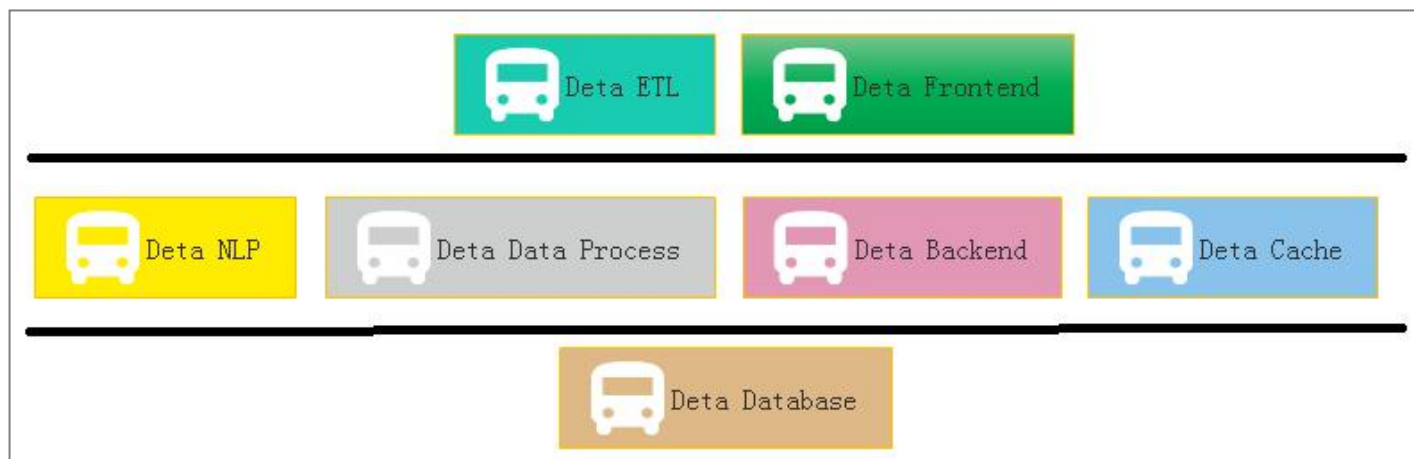

Figure 1-2 Sections of DETA Projects Group

Applications

One question is my friend asked me why does DETA support the e-commence logic? Definitely! Please see the Figure 1-3, this is a classic horizontal deployment sample of the real word. Alibaba, Amazon, Ebay and JD etc, all based on this technology, instead of Spring, DETA can be the next generation of technology.
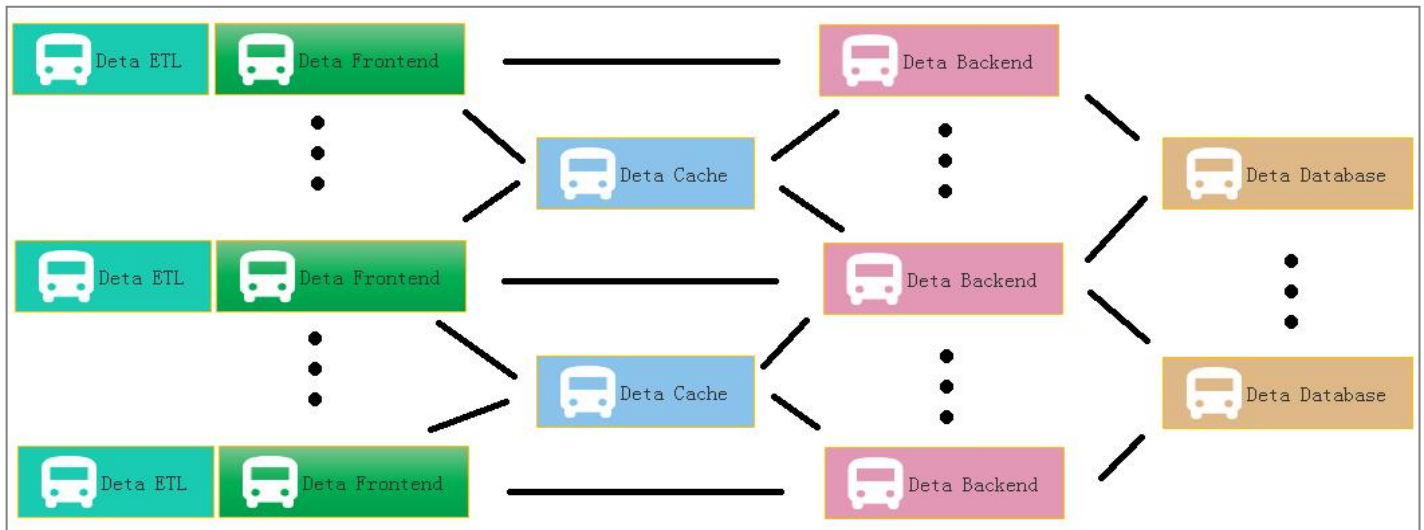
Figure 1-3 DETA WEB Projects System

At Figure 1.4 is a real sample for web Devops by using DETA Open Source.
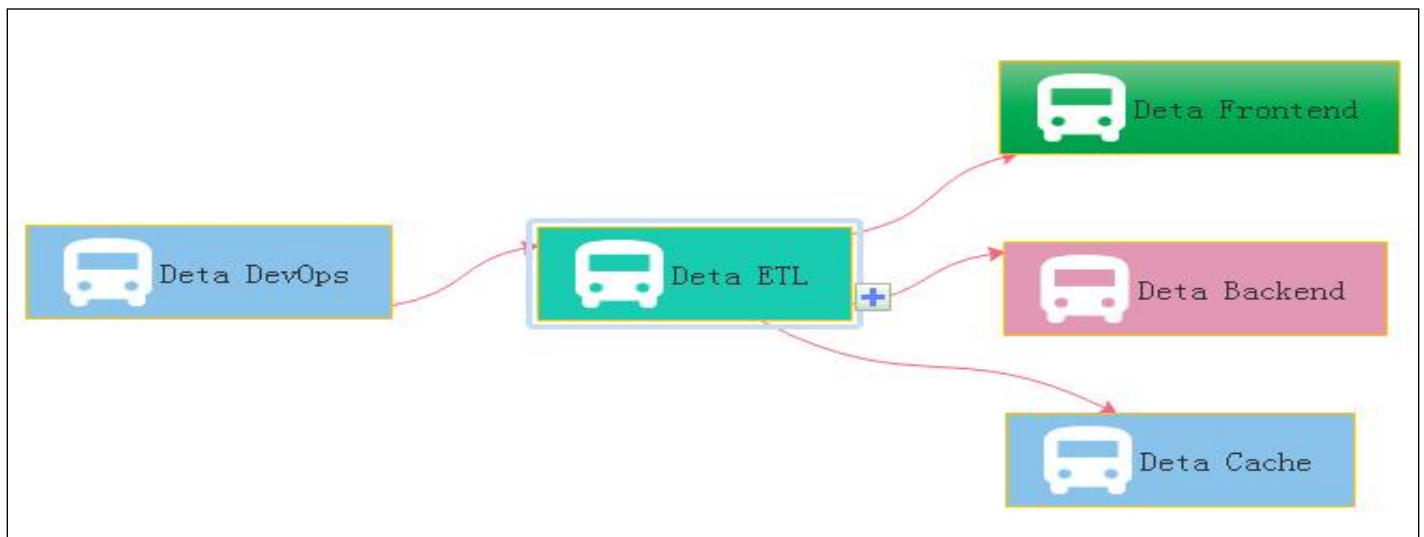

Figure 1-4 DETA DevOps Projects System

II DETA Business backend logic

Before 2010, the author systematically contacted the mechanism of analyzing A, operating O, processing P and managing M in the learning process. After graduation, he had the opportunity to deal with the business logic corresponding to these things through programming in some software companies in the society. From the research of MP6 mail system of French ESIEE University, The Bluetooth group advertisement machine to Shanghai World Expo, from the e-commerce back-end calculation of Yamibuy Company to the global hotel reservation of Ctrip, the author has been thinking about how wonderful it would be if the front-end system could give wisdom like human beings. So the bud in the author's heart began to take root, and he was confident to design a set of architecture system with humanoid wisdom to meet the rapid development of business intelligence applications.

II I DETA Business backend logic history

The first contact with the Frontend and Backend separation was in 2004, when the author first published a website in

Liuyang city by using the 7week platform. The website was a second-level domain name, using a third-party server, even though the concepts of front-end and back-end were ignorant at that time. The author first contacted MVC architecture in Shanghai Fanteng Information Technology Company. At that time, the feeling was that MVC could solve all kinds of business logic. In the same year, the author first came into contact with MVP to do multi-thread Bluetooth big file project, and felt that MVP seemed to make the architecture handle the problem of concurrent computing well. From 2014 to 2017, the author worked almost with business logic corresponding to various MVC architectures, such as Spring, Martini, etc. The author thinks that gives MVC an intelligence urgently.

II II DETA Business backend logic development

Thanks to my father, in 2018, he told me to design a pharmacy-assisted search software according to the concept of Chinese medicine, so he began to design Huaruiji Medical Big Data System. At that time, I thought spring boot, mysql were too heavy. If the database rest handshake system of socket stream was designed according to CGI, many problems could be solved easily. So I began to analyze, operate and deal with the problems. Gradually found some irreplaceable primitives, such as S static data, V visionary observation model, P procedure registration mode, C control unit, etc. It would be wonderful if we could redesign a set of architectures for these primitives. The PC separation mode here comes from an IOC doctoral design paper in Spring in 2015. Thanks here, I integrated MV into V observation model, and then took out the corresponding static data of M and function S. This VPCS structure choots me at present.

With regard to the excessive description of VPCS, I can take a previous note as follows: VPC architecture programming thought, software programming for many years, accumulated some thoughts on program realization. Through the certification of Darwin's theory of evolution, an effective VPC programming concept is elaborated based on the neutral coupling of MVC+MVP. V is an observer model, similar to storage object and observation model. P is the processor, which handles the registration interface. C is the control machine, which describes and classifies the registration interface. S static control machine, why use static control machine, advantages: 1. Because of the separation of PC, the functions of C mode are inherited through abstract virtual functions, interfaces inherit interfaces, interfaces are uniformly registered, and calls are extremely discrete, thus achieving the efficiency of high-speed concurrent iteration. 2. Realize EI separation and skip IOC scanning. 3: P is responsible for reference and description, and C can carry out various functional operations through descriptions of multiple P. Mapping control technology ensures thread safety and stability. 4: V stores each single case class to ensure low data redundancy and unified recovery.

II III DETA Business backend logic application

In the whole year of 2019, the VPCS back-end engine gradually formed some standardized functions and papers, which were applied to the front-end, back-end, cache, database and other subsystems of DETA. My evaluation of them is that they are lightweight and extensible. VPCS is gradually integrated into the works of Yangliaojing and Huaruiji. Of course, there are many shortcomings, the biggest one is that they do not repair themselves. Although I designed the sleeper and hallkeeper mechanisms, these mechanisms are only the corresponding business logic units that I complete through decision trees, not humanoid evolutionary thinking. At least, I don't think they are humanoid intelligence. To be precise, at present, they are only artificial intelligence, a kind of artificial intelligence logic corresponding to AOPM and VPCS, but not the humanoid evolutionary intelligence logic that I want. So I started to explore humanoid computing again. About the application principle description of VPCS, the author designed a paper as below:

VPCS Backend Theory And Its Application

Mr. Yaoguang. Luo

Liu Yang DETA Software Development Limited Company, Hunan, China,

313699483@qq.com

Outline: due to the development of the software acquisition and definition in what we use the code theory always in messy and unforeseeable status. A new method of the coding style like VPCS that will show in this topic paper, feel free to resonate with my imagination of the portrait—VPCS(Vision, Process, Controller, Sets) theory, fun yet? Not only this paper will gazer a big point how we show the constructions of the VPCS, you guys also sure to get lots of idyllic landscapes of the coding sections. While you got lots of the illness codes at the so messy fungus projects, I guess at this paper out where you are finding anxiously. Let's catch more opportunity about how does the VPCS working, executing and scheduling in our software project and make the software fast, fast and safe! lets go, So the key words as below:

Quantum Sets, Concurrent Consumer, Vision, Scheduler, Threads, Surf.

Introductions

Let see the verbal keys, the first time you …, okay, get any sense? Sure, this paper is not talking about the human careers, truly about SOFTWARER, as a human, if you got my points, yes, cool! Make any sense? Let's see the landscape as below figure 1-1.
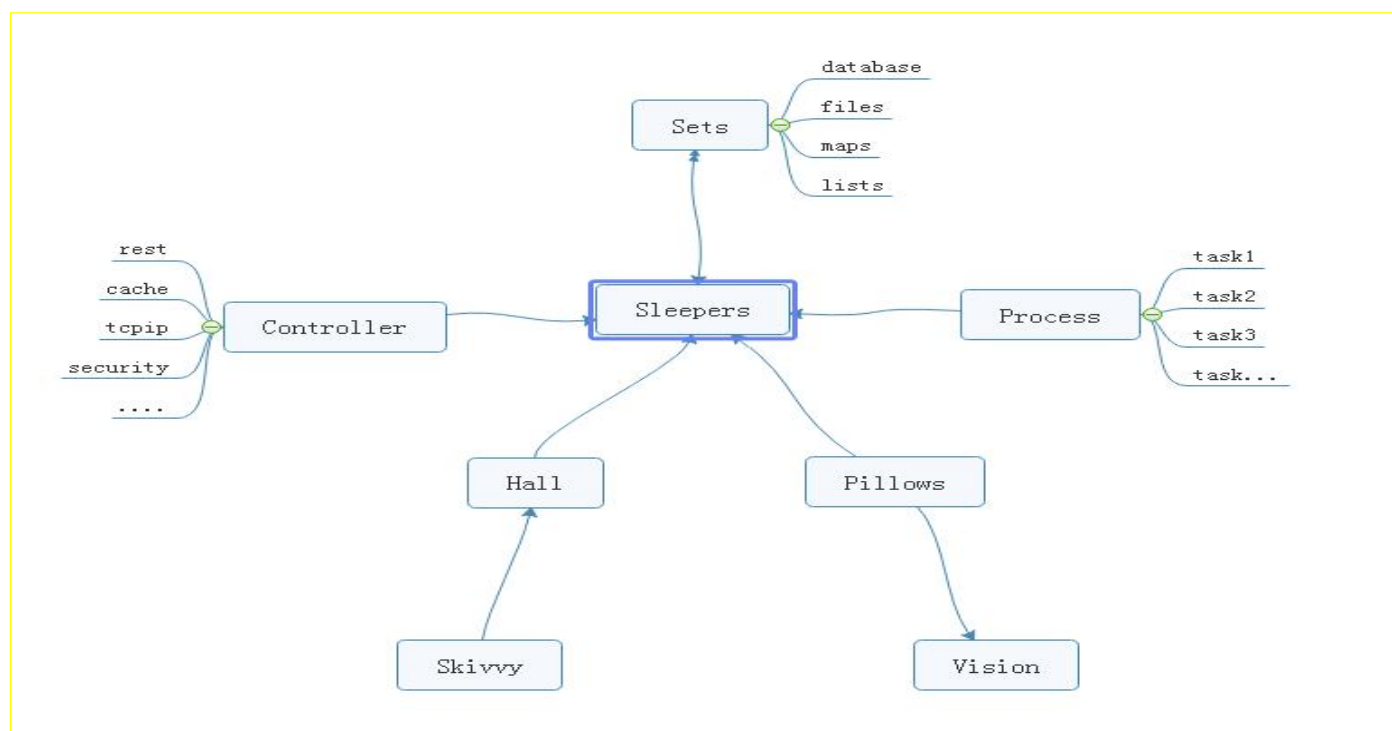

Figure 1-1 VPCS STAR MODEL

From the ordinary software development architecture, always like a factory model, for instance, controller, transaction delegate, web service, job bean, data DAO, like that of traditional backend or front end coding style, but, compare now the seamless clients services system, those model more and more not suitable for us for the project application, at least in the light level, multitasks, satellite boots projects system, if we choice the factory model, you will feel so heavy. But the big conflict problem is where the factory model was used in all and all bazaar companies. Even more CTOS that I met before often complaining about the reference room likes that "we need one server for database system, one more for cache system, one more for front end, one more, for backend, one more…."after that what do you think? My lord…

Finding a new method of how to integrate the sets about the micro satellites service in the same sever, and make them small, lightly and faster for the commence service, now become a fatal topic. Which can be a pretty warm-up for where I make

an explanation for VPCS. The VPCS model, only includes four aspects. Vision, Process, Controller, Sets, and those factors makes an interactions in the sleeper containers. Let talk about the definition of the sleepers. From the software engineering domain, the sleepers are more like an identified thread person. Who can make a lot of fantasy dream in a Hall, what means a dream? Dream is a requirement what the consumer really needs to finished. But here the dream can be separated out more tasks, those tasks will register the ID in the Pillow, so that the sleeper hugs the pillow then goes into the hall and make a dream. Got an idea? Cool. So what does the sleeper does in a hall? The answer is to make all kinds of the dream. For example if we want to build the web service to get rest call, and return the JSON feedbacks, we only need to do like the way: Firth, build rest call path in the controller; Second: register the call requirements as a dream; Third, build the sets of the dream in the pillow, Fourth hire a sleeper to hug this pillow, and go to the hall to make a dream process. At last but least: return the dream goods. Any sense? Cool! For this unique instance, you will know that the sleeper was more like a socket, and the hall more like a thread pool, the pillows like the single vision instance, and the sets like a vision storage, the controller and the process those two sections is a common way of the factory model. The steps landscape of the sleeper who makes a dream as bellow figure 1-2.
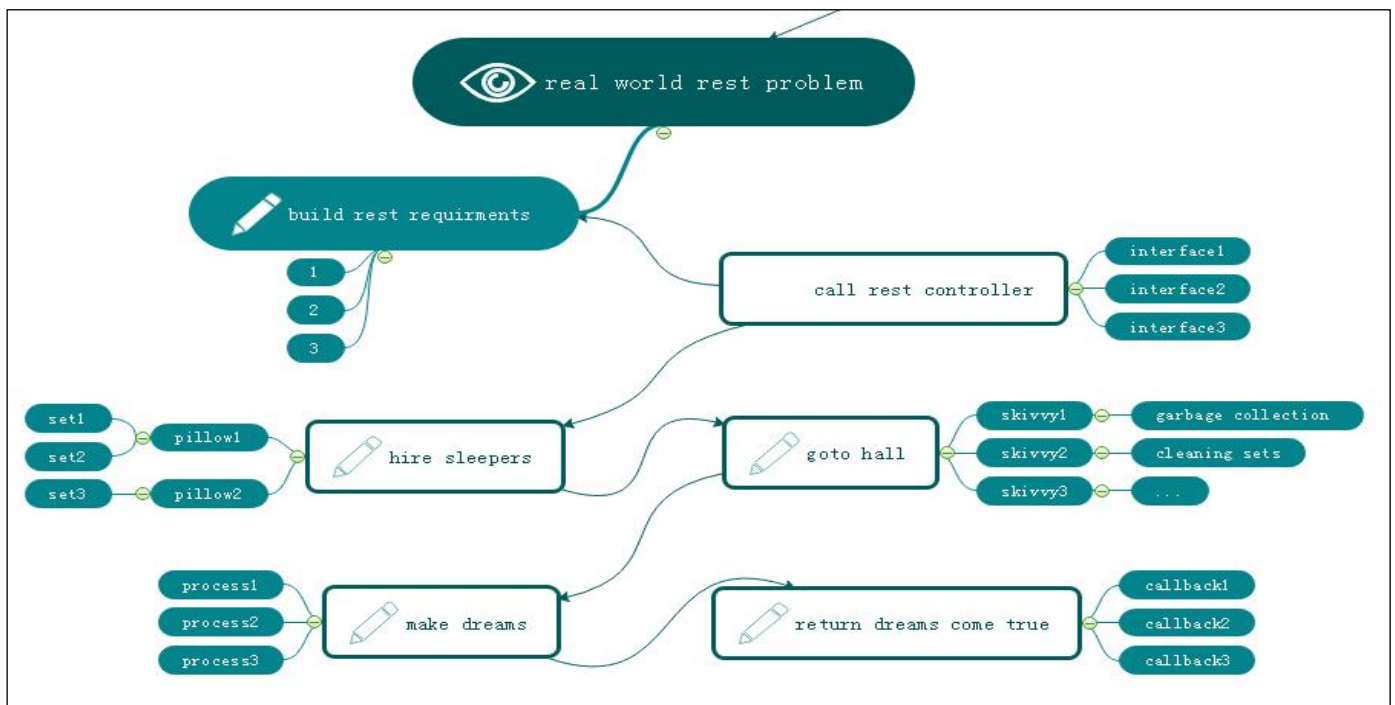

Figure 1-2 VPCS BACKEND MODEL

Focus on this landscape, mostly different to the MVC: Model View Controller, MVP: Model View Presenter or other architectures we know before. but is very easy to understand after you read for a while. Too simple. Sleeper makes dreams come true, hall container sleepers, skivvy make up the hall, pillow clear and wake up the sleepers who often lost in finding the way in the dream. Got fun here, but I would hear more argue voice DETAils of my VPCS, desktop App once said: VPCS is good in the concurrent WEB project, but not suitable for the desktop applications. Ok, follow this question, let make a new landscape based on desktop application as below figure 1-3.
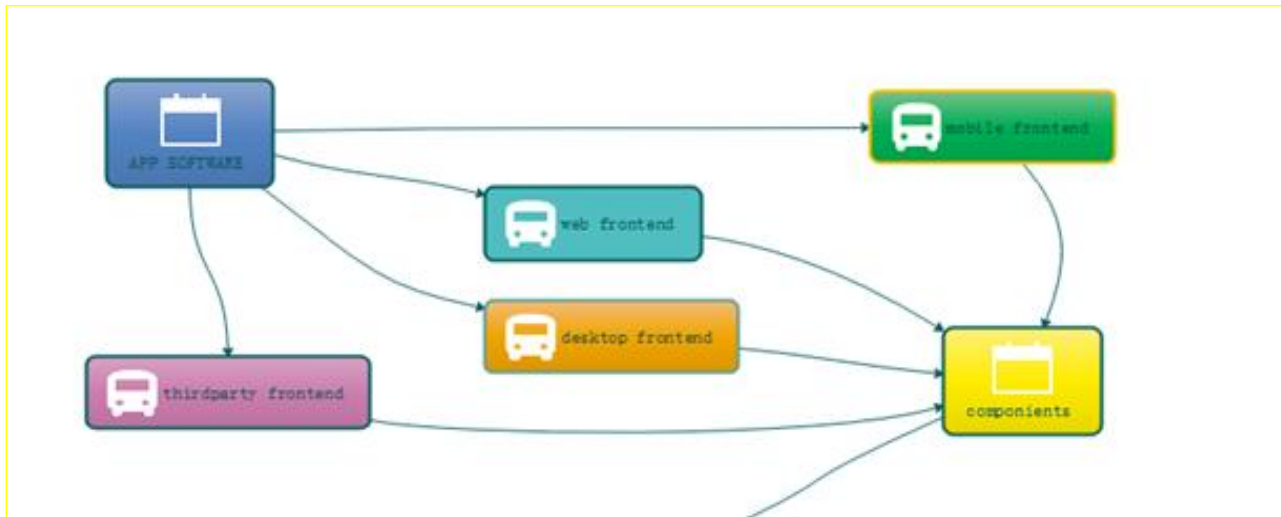
Figure 1-3 VPCS WORK WITH FRONTEND

From this picture, we know all of the software can be fast and safe while using VPCS, because it is already separated out the big system into backend and frontend two parts. and VPCS keeps safe and fast in the backend section. Compare to the MVC, VPCS will get more cautious and DETAils, and compare to MVP, VPCS also will get more safe and high efficiency. Those factors are why I will make inauguration here. In the common software engineering cycle life times, scientist used to build front end and backend for all kinds of the software applications, because it is easy to control. Why? Frontend only spend time to make design, and Backend for the data operations. Using VPCS system, we don't care about what they do for the front end, we only fit about what they want. Alignment that gets a blame and fix, then return OK, the restful service developer makes a voice that http functions are concurrent functions. At here, VPCS will say: concurrent functions are safe functions. We guess in the future REST-VPCS will be used in multiple WEB service. Especially in the high speed, efficiency, micro web systems with high level security for example medicine, DNA, cloud server, electronic police system and ecommerce systems etc.
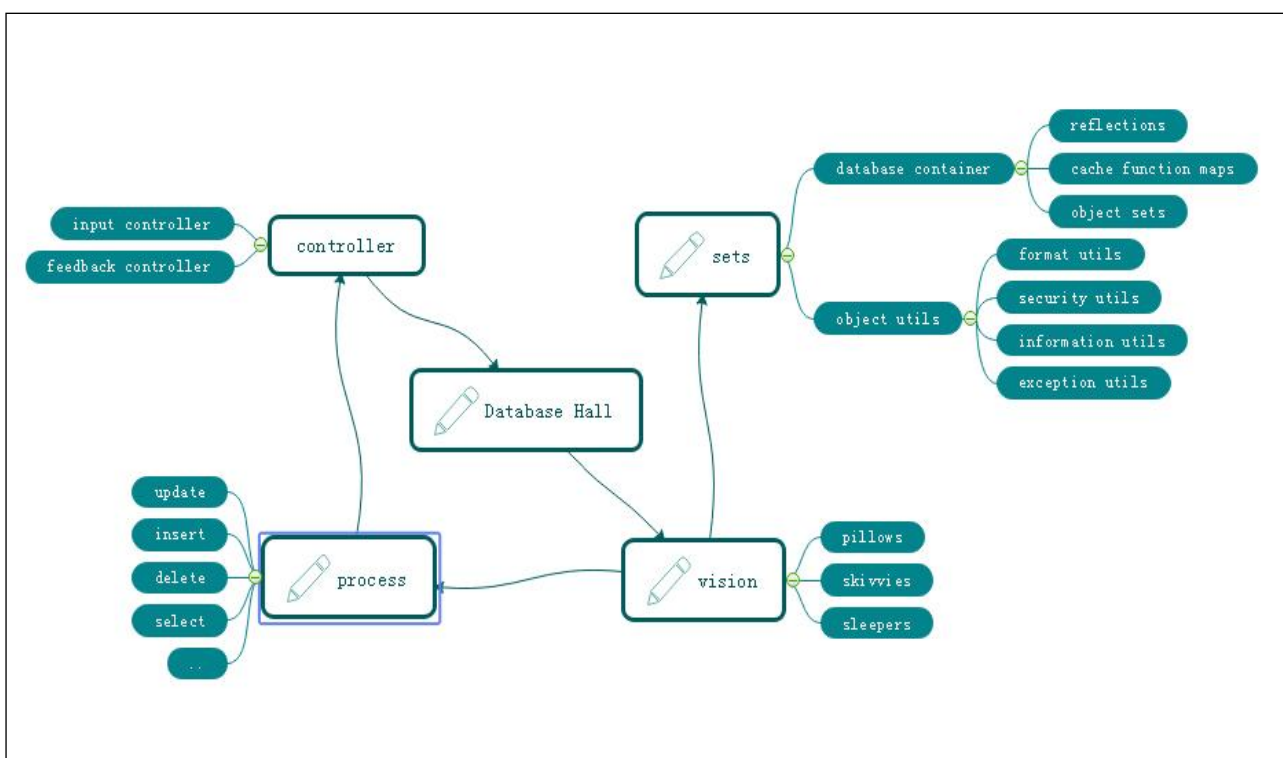
Figure 1-4 VPCS FOR DATABASE SYSTEM

As the figure 1-4, a new method of the Database system designing shows us VPCS is a pretty way for the modern data information management system. Definitely used in the DETA Database system. For this instance, do we get a view that the controller section of the factory model becomes thin yet? Controller only works for the hands transactions, for example that the controller get an input requirement such as select SQL, then immediately call the hall keeper to register this SQL and hire new sleepers to make a result. Because of the VPCS. Once it happened any exceptions, will very easy to awake sleeper and let them get theirs working papers out, finally call skivvy to fork the sets to the fresh sleeper. This method mostly be like a Count Down Latch model, once the sleeper gets the dreams come true, then told the hall keeper for the feedback, hall keeper will makes a type procession to return after everything goes well, This method mostly be like a Cyclic Barrier model.

Questions

How does skivvy doing? please see the figure 1-5 the hall building need a singler instance like a home keeper but here is a hall keeper, any else, this person is very important for keeping the VPCS safe, because all of the skivvies will be managed by him. You will see, the memory check, JVM garbage collection, disk cleaning, thread status management, deadlock alarm, security protocol all and all in one at here. Mostly like a static class in the VPCS system. If we need to know every thing about skivvy's work status, ok just call the hall keeper.
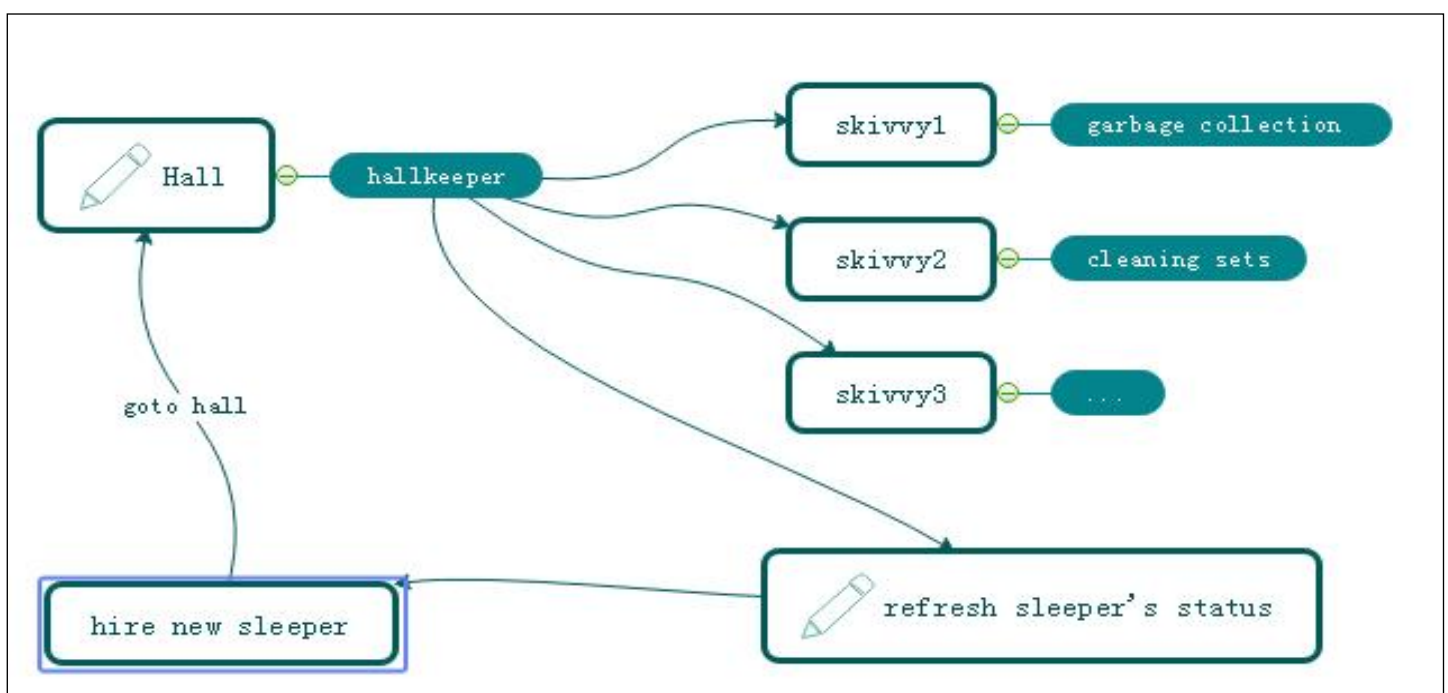

Figure 1-5 VPCS KERNEL

How does sleeper doing? Make a dream? Cool, you shoot!, in the VPCS system, it doesn't have the definition of the process, everything likes subsets. Immutable or unlined, hall keeper get request from visionary and hire the sleeper, who is likes a thread, get requirement, add those sets in pillow, hugs pillow then go to hall to make a dream, after that then return the callback to hall keeper what they did. Fun yet?

What does the sets meaning? Sets, is a format of the data where appearing in the VPCS system. For the static prototype, it used like a concurrent hash table, and list which can be copy base on writing format, the single instance, it always runs in the static function or be liking an interface implementation because need safe at the same time, so that compare to the factory

model, it is too simple and without annotation. Everything becomes easy in this environment.

The one more question is that so many peoples asked me what does the sequence diagram of the VPCS, because they really want to know why VPCS is faster and safe. Ok, please see the figure 1-6, the answer is absolutely, VPCS main components of the time sequence only contains five aspects. Almost similar like the hotel management. Certainly, we are talking about VPCS software, not for guesthouse. You will see that the rest call only makes the interactions with the hall keeper. And hall keeper got two jobs, one for waiting the fresh sleeper and one more for giving task to skivvy. The sleeper only hugs the relate pillow and make the dreams come true. Fun yet? Cool. VPCS only take cares about how does the sleeper's imagination and skivvy's working status. If is the pillow broken? Make new pillow, got lazy sleeper? Get out his working papers, got a cheat skivvy? Fix of fire him, the real source of the java version project for the VPCS only 30kb, we will find more sources or documents from the reference links at the end.
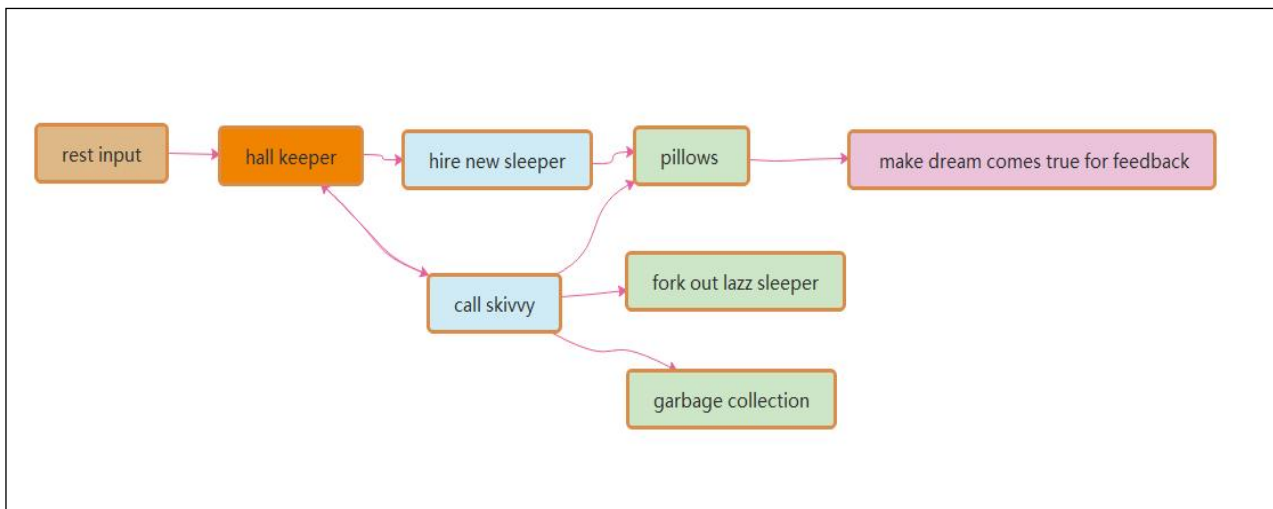
Figure 1-6 VPCS Sequence Diagram

I always be asked by the colleagues that once said: how does the hall build? I answered them, such like the hospital, no one cares about the address of hospital, because they just call the cell phone number when will get a directly feedback. This is why I need a hall keeper role in the gate way. For the instance about figure 1.6.1, this sample is a true demo in the real world for the WEB rest service. Its very important to create a player role such like hall keeper. what would likes about author's theory? Because of the maintenance. Because of whom, the software build team are very easy to make a maintenance web portal, all of the system current status will be solved on this html page by DEVOPS.
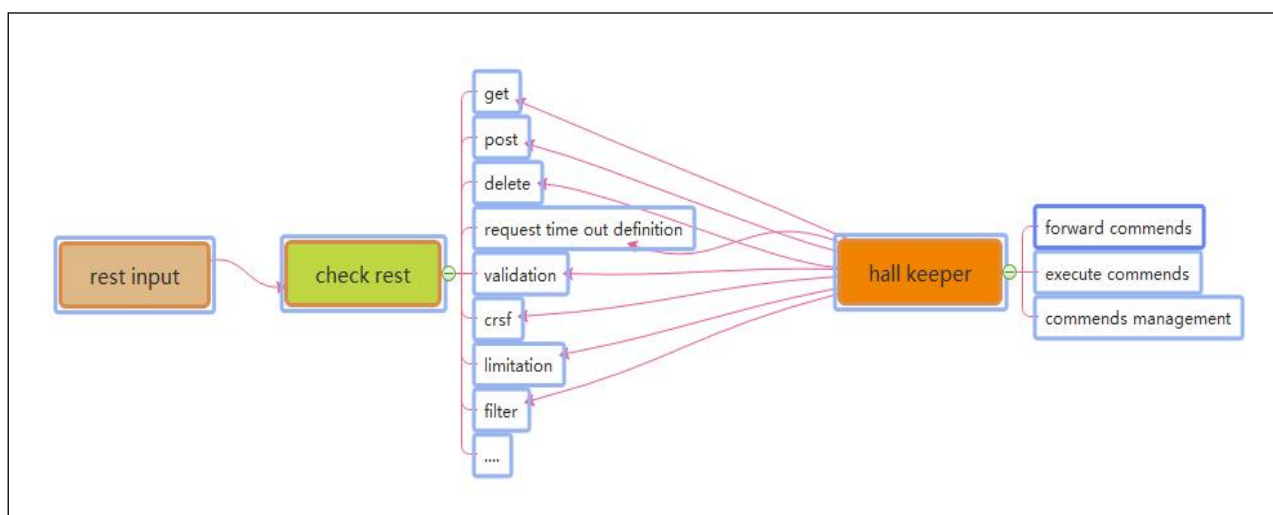
Figure 1-6-1 The Interaction Between Rest Call and Hall Keeper
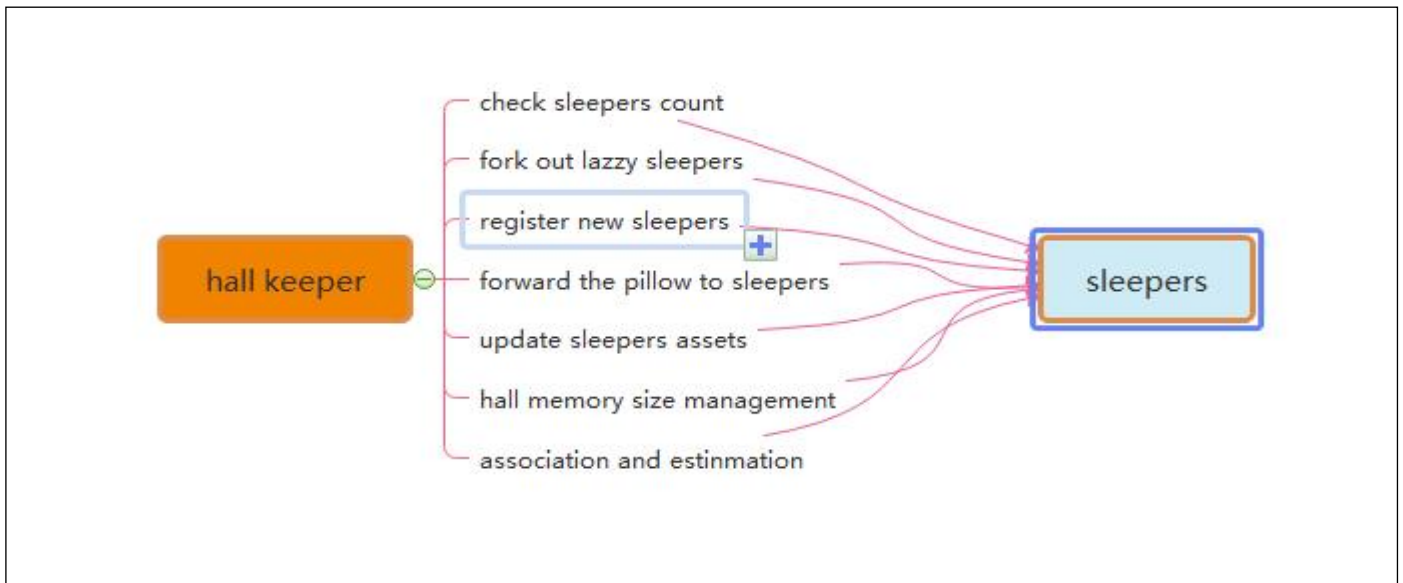
Figure 1-6-2 The Interaction Between Hall Keeper and Sleepers

Many of these software developer also asked me how and why we fork out the LAZZY sleepers excluding their sets. Arthur answered because of the pillows. When the sleepers be hired from the hall keeper, they will get an independently pillows such like static functions. So that sleeper only has their own indentify attributes and unique information as the singe instance class. Once they got theirs working paper, the pillows they used will be arranged to the new fresh sleeper, this theory keeps safe, quality and quantity. Like figure 1.6.2.1 VPCS kernel.

Figure 1-6-2-1 VPCS kernel

Figure 1-6-3 The Interaction Between Sleepers and skivvy

Many times, I got questions about the DEVOPS, they really worry about VPCS if suitable for their project system maintenance? The answer is absolutely, as the Mr. Ray 274138705@qq.com once said: we are DEVOPS, at least we need three important keys in our environment assignments: implementation capacity, transparency and maneuverability. How does the VPCS supports us for daily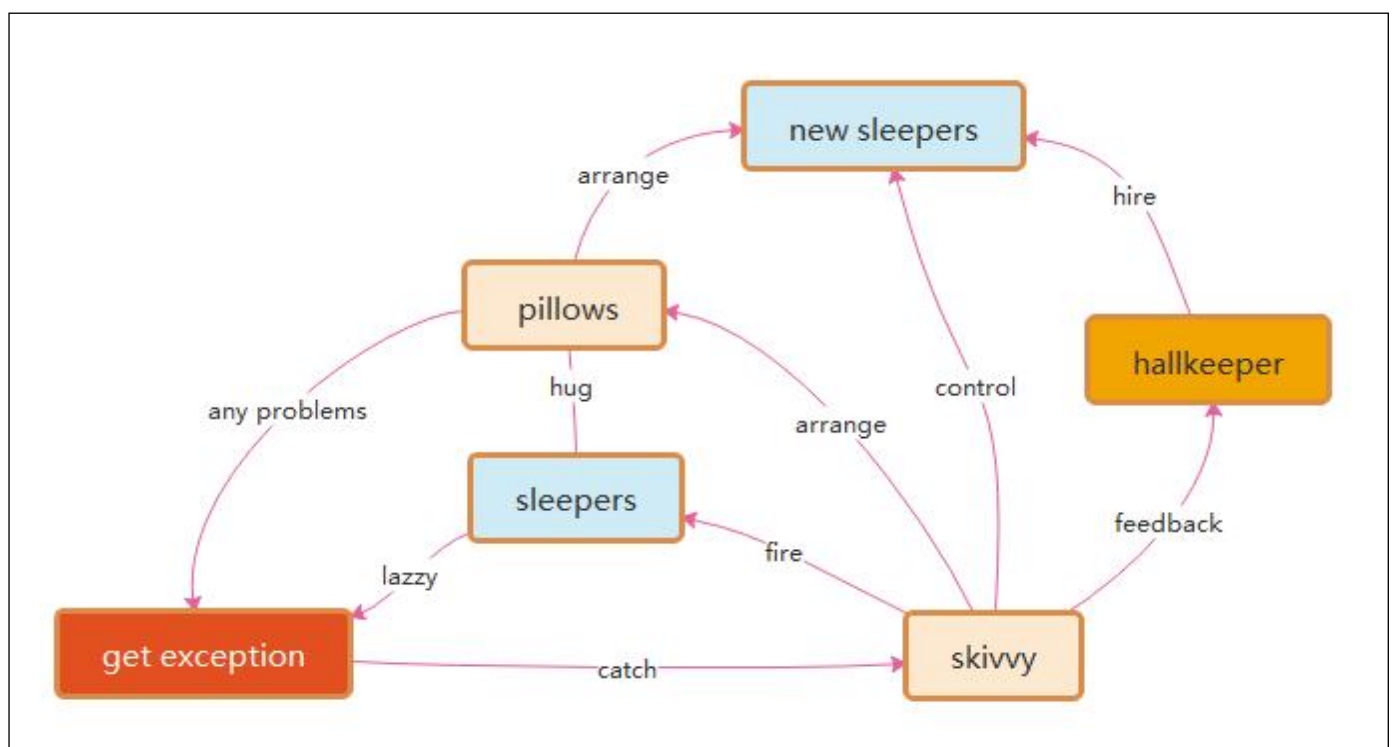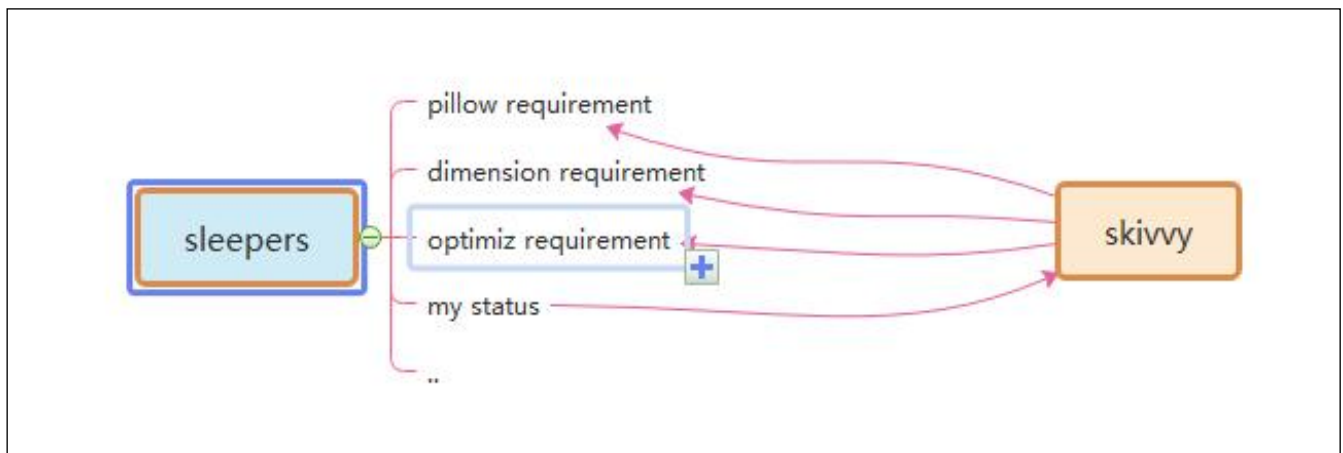 works? Because of Hall Keeper and skivvy, as figure 1-6-3 and 1-6-4. DEVOPS will get all of these transparency information about project from hall keeper under the encryption and security issues. Also, hall keeper will directly get the rule for DEVOPS by rest calls, then makes to commend to skivvy. All of the information and record logger will be cached by hall keeper, that keeps controllability. The html control page will make an interaction between hall keeper and DEVEOPS, which keeps safe, implementation capacity, transparency and maneuverability. This is my true answer.
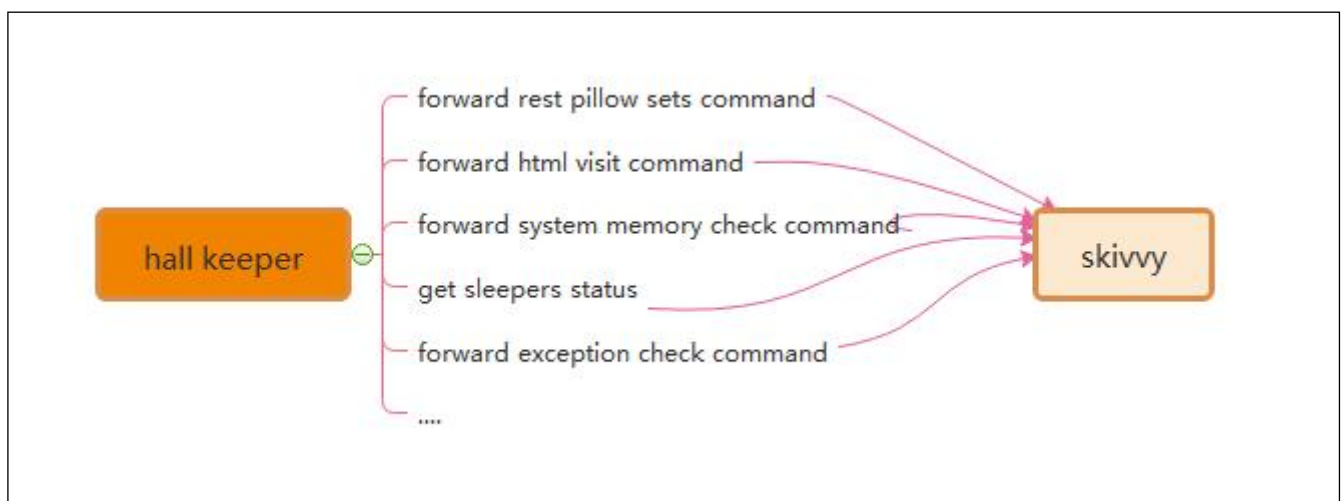
Figure 1-6-4 The Interaction Between Skivvy and Hall Keeper

Recently Mr. Yang 1291244774@qq.com who asked me about VPCS of IOS desktop APP, where and how to avoid the data leakage risks. Because he really worries about the separation between controller and process. Following this topic, my answer that the key is the separation between pillows and sleepers. Due to the pillows all have their own unique ID, skivvy will easily arrange the pillow to new sleeper after the original sleeper who made problems. Make unique ID and arrangement by ID, is the key method. Also for the rest call service, the asymmetrically irreversible combination encryption is one of the best solutions to the data leakage controller. VPCS seems so smart.

Figure 1-6-5 The Interaction Between Sleepers and Pillows

III DETA Catalytic computing

I got my mind storm for a month in early 2019. How to realize human computing? It's been bothering me for a long time. How to start? I didn't have a clue, so I started to read my notes made in the past 20 years. I got it! Do basic research! According to my notes, I dig some unknown basic knowledge. I am very happy, because I have the results of quick sorting by left-right comparison in 2014, the butterfly calculation manuscript of Fast Fourier, the Chinese word segmentation works of Huaruiji, UNICORN ETL, socket stream PLSQL database, etc. and an idea came into being. I think about optimizing them continuously, refining, optimizing and testing repeatedly, and remembering these optimized ideas.
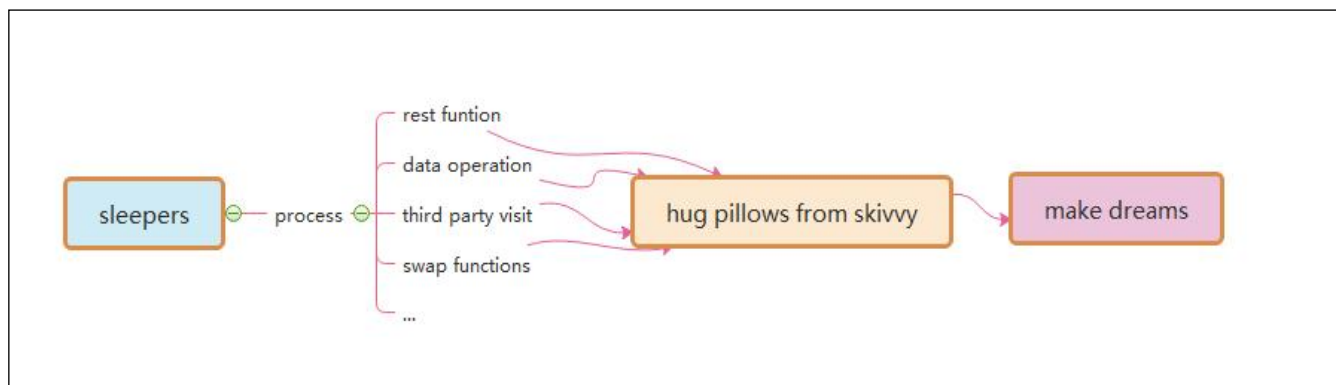
III I DETA Catalytic computing history

Since of the bright flashes, may I follow the operation method. The refinement method of DETA's first catalytic calculation is first reflected in DETA parser, such as the refinement of semantic part-of-speech analysis, the optimization of flow valve, the irrational conditional transformation of discrete data, the filtering of the same frequency operator, and the filtering of calculation peaks. These optimization methods of human thinking gradually form a system, which not only changes the design mode of DETA's works, but also changes the author's research and development philosophy.

III II DETA Catalytic computing development

R&D is not successful every time. In the process of butterfly calculation optimization of Fast Fourier, I coded the features of discrete DCT in complex numbers, which took me one month, but failed. I remembered that I said in Weibo at the beginning that I could speed up the calculation of Fast Fourier by 200 times, but I really didn't give up. Since butterfly calculation optimization was unsuccessful, I tried to sort the small peaks by fast left-right comparison. I was excited when I saw the 10th generation of single machine random double with a sorting speed of 12 million arrays per second of quick sorting. My thought is right, and thinning logic is an important way of human thinking. Here, the author designed an argumentation paper when designing fast word segmentation and extremely fast peak filtering catalytic sorting, as follows:

Theory on YAOGUANG's Array Split Peak Defect

Mr. Yaoguang. Luo
Liu yang DETA software development limited company, hunan,china
313699483@qq.com

Outline：In the common software development factory, engineer always did more and more interactions with data structure and math algorithms. Especially in the recursion, convolution, sort and generic loops, scientist likes to find a simple,

more sufficiently and alignment way to face the project requirements with the large association. For instance me, I really got a real world problem at this domain while I use quicksort, also for other project such like DETA parser. What is the peak array split defect? How does it count the real world problems? Why need find it and how to get the nice solution? Cool, this paper will cause an implementation about our goals, ok now, keep forward to the context where I talking as below, thanks For more theory DETAils and the source code implementations please check the bottom reference section.

Peak, Array, Split, Defect, Recursion, Convolution, Sort, Generic

Goal one: Quicksort Yaoguang.Luo 4D

1 DETAils：

1.1 For example the array input as below where we gave 11 digits.
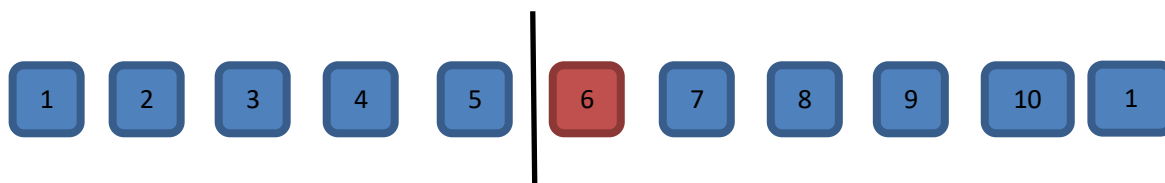
1.2 The first split, we could see the digit-6 will auto arranged to the right part.
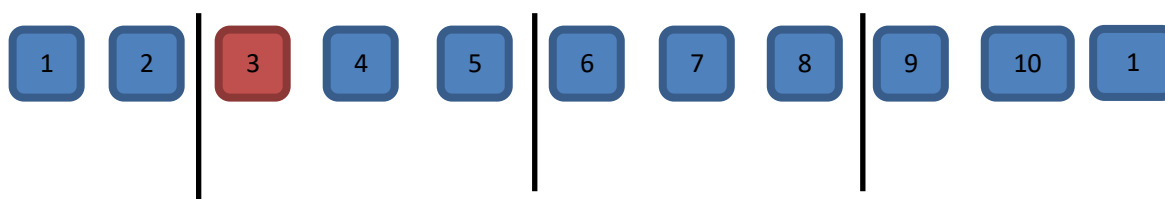
1.3 And the second split, we may see the digit-3 will be auto arranged to the right part
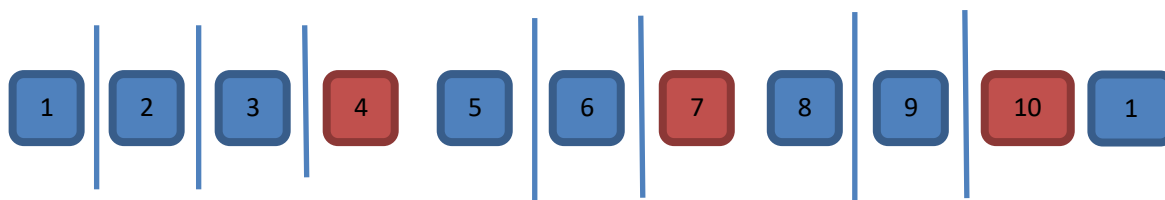
1.4 The third split, we may see the digit-4.7.10 will be auto arranged to the right part

2 Thinking:

   After the split array showing, we could see clear that the big problem about the asymmetry defect, as I did an annotation of N, so the i of N will absolutely find a n/POW（2，i）value points, as an insufficiency asymmetry defect model, I fall in thinking…if I do any compute theory as the same with this model style, for example in the recursion or inner loops, it will autonomic separate to the 2 different process way, it necessary to do indifferent flows.

3 Problems:

So, after the above thoughts, I may get any flashes, First, the even and odd digits both are asymmetry while in the Differential loops. For this noise, I defined as (Tinoise Peak) Second，once we did a split compute under this model, it must get more unfair sets. I defined as (Tinsets defect). Third, if this model almost in the messy and timer data system, it will catch more time and asserts wastes or exceptions.

4 Solutions:

For the god like, I find three solutions while I currently enrolled in my projects. First: computer logic acceleration, at least it can avoid the waste of the compute by using inner process optimism. -- To avoid the deep recursion. Second, reduce the compute sets. For any less memory system, we may reduce more and more memory garbages after we reduce the inner register or temp value sets. Third, we may make an optimization of the function logic where to instead the old complex functions. Those ways include the condition, algorithm, method or discrete optimization. End, we may use mathematics of double differential, deep definition, acquisition or polynomial to get the solutions.

5 True Instances

Let me show the algorithms here,

```
private int partition(int[] a, int lp, int rp) {
        int x=a[lp]<a[rp]? a[lp]: a[rp]; //reduce the compute values, reduce the recursion peak
        int lp1=lp;
        while(lp1++<rp){
                while(!(a[lp1++]>x|| lp1>rp)) {// reduce the condition differential check, reduce the recursion loops
                }
                while(a[rp++]>x){
                }
                if(-- lp1<--rp){
                        int temp=a[rp]; a[rp]=a[lp1]; a[lp1]=temp;
                }
        }
        a[lp]=a[rp]; a[rp]=x;
        return rp;
}
```

From this code: in a common quicksort way, the recursion based on the average deep split, suppose the initial array length is N{1,2,3...n} is an Odd, so the separate two arrays will cause an asymmetry defect, those timer asymmetry compute peak collection will cause more and more probability problems such like jam, lock, time waste and heap increment.

The odd peak binary split as below：

Defect peak
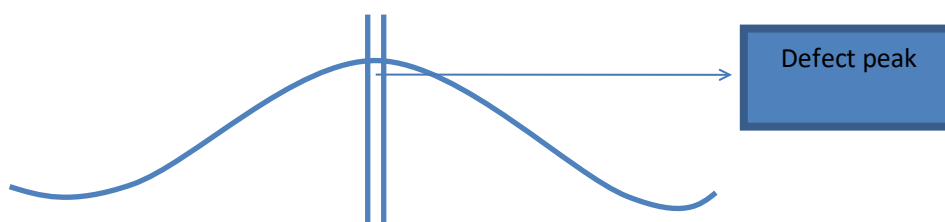
How to avoid those timer distinction peaks? I go more absolutely research where focus on these problems, first, differential flows. This flash is not suitable for here, May good for the DETA parser, I will show you later. Second, compute acceleration. Yep, this is a good way, for example find the big X as the code blew, it will cause the while loop ability accelerations.

<p style="color:red; text-align:center">*int x= a[lp]< a[rp]? a[lp]: a[rp];*</p>

Third, De Morgan condition differential as the code below, it will cause the condition ability accelerations.

<p style="color:red; text-align:center">*while(!(a[lp1]>x|| lp1>=rp)) {*</p>

At last but the least, value reduce, code optimization both are very important way of the peak avoid filter.

Goal Two: DETA parser

6 DETAils：

Last year I help my father to develop the study software about getting the medicine data collection for quick search. I' m going to try to build a search engine system, input format is a string, how to get a Chinese string array split?

| 香 | 蕉 | 和 | 苹 | 果 | 很 | 美 | 味 |

6.1 convolution length indicate by marching Nero index tree as below: 2|1|2|3

| 香 | 蕉 | 和 | 苹 | 果 | 很 | 美 | 味 |
| | | | | | | | 很 |

6.2 convolution POS indicate as below: n |c |n |adv |adj

| 香 | 蕉 | 和 | 苹 | 果 | 很 | 美 | 味 |
| | | | | | | | 很 |

6.3 convolution split

| 香蕉 | 和 | 苹果 | 很 | 美味 |

7 Thinks:

    While I use this way on my DETA project Chinese separations, I met so many problems, the more and more important problem is the POS frequency peak waste, my POS flow functions will spend a lot of time to do the low prior convolution split condition check first… it cost me a lot of time… after I did a collection of my projects, the results are clear.

8 Problems:

    First, convolution kernel gets asymmetry problems. Second, unnecessary conditions check loops. Third, unimportant heap register values. End, values sets and conditions sets too more.

9 Solutions:

    First, format the convolution kernel of the index dictionary tree, for example I use the char ASCII as the index length to reduce the match time of the convolution length indication. Also, I define the POS convolution kernel size less than five.

Second, I did a condition frequency statistic, and re arrangement it, at the same time, reduced a lot of the inner sets to avoid the compute pause.

10 True instance:
Finally I developed a convolution String array split way for marching as below: orange color are presplit sets

10.1 check 4 chars slang

4	4	4	4		

10.2 Check 3 chars key word

3	3	3			
	3	3	3		
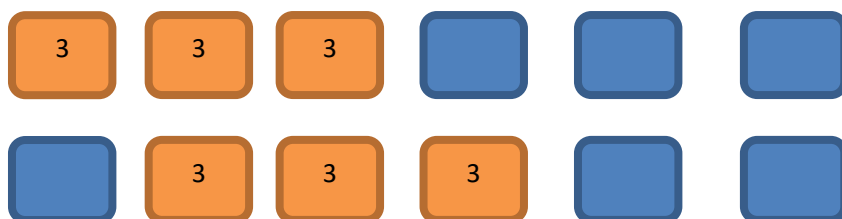
10.3 Check 2chars normal word

	2	2			
2	2				
1					
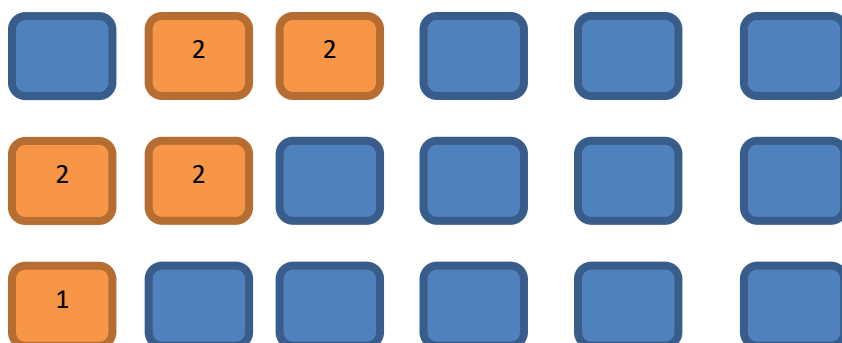
最后单词拆分
10.4 In order to make a compute acceleration, I did 2 string builder array to store a pre order sets.

10.5 in order to make a PCA POS acceleration, I did 5 chars marching array to store a post order sets.

It seems the Nero Index, NLP and POS for the PCA separation with convolution kernel marching by using stepwise iterative differentiation got much higher sufficiency.
计算拆分的高效性可以使用我的德塔分词的作品测试. 充分的数据论证其性能是优秀的.

III III DETA Catalytic computing application

In order to demonstrate the importance of DETA iled logic, I began to integrate this logic concept into my YangLiaoJing and all my soft works. When I saw 13 million high-accuracy word segmentation per second, 6 million mixed phonetic symbols per second, 12 million double arrays sorting per second, and other amazing works came out, I began to sigh my own cognition. I unreservedly opened up all these ideas and works, which hope aroused humanoid thinking resonance.

At present, the significance of differential catalysis has included seven categories: frequency valve (von Neumann) differential, discrete logic (De Morgan) differential, high-frequency function degradation, conditional refinement differential,

executive mode differential, giant system (Qian Xuesen) module differential, and mathematical differential (Newton, Blainez), which is the only way to catalyze humanoid DNA evolutionary algorithm. And made a DETAiled demonstration and summarized as follows

DETA Human-oid DNA Details

1

https://gitee.com/DETAChina/DETAParser/blob/master/wordSegment/org/tinos/engine/pos/imp/POSControllerImp.java

Demonstration of differential algorithm of POS water valve with DETA fast segmentation;
Demonstration topic: Differential catalytic calculation can well observe the execution flow logic of quantum state function. Through statistics, high-frequency function can be advanced gradually according to von Neumann state, and low-frequency logic can be eliminated and screened gradually.
Demonstration result: The demonstration was successful. Reduce the traversal times of irrelevant code. Greatly improve the computing power.
Demonstrating influence: the main way of evolutionary mechanism of humanoid DNA evolutionary algorithm.

2

https://gitee.com/DETAChina/DataSwap/blob/dceeb0b06f726d640553964058d85b736354ac89/src/org/DETA/tinos/array/LYG 4DWithDoubleQuickSort4D.java

Demonstration of The 4th generation example demonstration of filtering sorting algorithm for Luo yaoguang s small peak calculation;
Demonstration topic: Differential catalytic calculation can be combined differentially with discrete mathematics system to filter high-frequency functions where from the digital logic level. Ensure smoothness
Demonstration result: The demonstration was successful. Greatly increase the calculation speed.
Demonstration influence: humanoid DNA evolutionary algorithm can effectively smooth the peak of computation.

3

https://gitee.com/DETAChina/Data_Prediction/blob/master/src/org/tinos/DETA/tsp/YaoguangLuoEulerRingTSP2D.java

Demonstration of the second generation of differential TSP algorithm for Luoyaoguang Euler forest business travel ring
Demonstration topic: Differential catalytic computing can optimize the thinking mode of traditional complex logic at cognitive level, and fundamentally change the cognitive process from the beginning.
Demonstration result: The demonstration was successful. Differential catalytic computing can change the traditional cognitive style in some social fields.
Impact of demonstration: Humanoid DNA evolutionary algorithm can effectively select the fastest algorithm module and cognitive module to do calculation in specific fields, and improve computing power.

4

https://gitee.com/DETAChina/DataSwap/blob/master/src/org/DETA/tinos/string/LYG4DWithChineseMixStringSort7D.java

Demonstration of the 7th generation example of Luo Yao s conditional differential sorting algorithm for light image strings;
Argument topic: Differential catalytic computing can unify conditional functions considerably, reduce logic complexity, and continuously optimize and focus.
Demonstration result: The demonstration was successful. Differential catalysis algorithm can split the local modules of the whole function by vpcs logic, and form the purine element of initon operation of DNA peptide chain.
Demonstration influence: the guarantee of autonomous evolution mechanism of humanoid DNA evolutionary algorithm.

5

https://gitee.com/DETAChina/DETA_PLSQL_DB/blob/master/java/org/lyg/db/plsql/imp/ExecPLSQLImp.java

Demonstration of PLSQL differential compiler for DETA Socket stream programmable database engine;
Demonstration topic: Availability of multi-condition execution of differential catalytic calculation.
Demonstration result: The demonstration was successful. Differential catalysis algorithm can provide reverse observable operation and maintenance guarantee for functional system operation.
Demonstration influence: the comprehensive application practice of conditional differentiation, logical differentiation, high-frequency valve preposition and other functions of humanoid DNA evolutionary algorithm.

These arguments were a year ago. At present, many works have been in a good follow-up state because they are developed as subsystems of the project of YangLiaoJing. Over the years, I have been thinking, what is the final expression of DETAiled logic? I have never stopped exploring, and I have always been absolutely focused.

IV DETA C Finding initions

I have been thinking, what is the final expression of DETAiled logic? I have never stopped exploring, and I have always been absolutely focused. I must find the final expression of these logics. From 2018 to 2019, I thought that the final expression of logic refinement must not be as simple as AOPM and VPCS. VPCS is just a refinement layer of AOPM, so how can VPCS be refined? So I began to sort out my existing things, my works and soft thoughts. God, I can only make persistent and absolute focus on what I have. I have to make a bet.

IV I DETA C Finding initions history

What's under VPCS? What is the essence of a function? At school, I got some basic answers. the primitives of DNA are ACGTU purine and pyrimidine, the primitives of back-end architecture are VPCS, the primitives of thing logic are AOPM, and the primitives of database are IDUC addition, deletion and modification. the primitives of function are IOAON Input, Output, And, Or , Negation, which I can only find in the knowledge structure I can understand and have. how to demonstrate? How to confirm the argument?

IV II DETA C Finding initions development

The first demonstration process is DETA word segmentation. In 2019, I continued to refine, optimize and refine the word segmentation, and found an exciting argument. My word segmentation function was continuously split rationally. Finally, a pile of simple combination application fragments of addition, deletion and modification were displayed by IOAON. For the most powerful argument, when I was processing nouns in word segmentation, the final function was formed. Memory takes out 4 words, compares 4 words proverbs, does not? then compare 3 words, does not? then compare 2 words, and does not? then split into single words. This process is summarized in one sentence as a combined decision-making process of adding, deleting, modifying and checking memory data according to John von Neumann's time flow form.

When I think about this, my eyes shine. IDUC is not only the operation mode of database, but also the operation mode of memory data, and it is absolutely focused continuously! ! ! Assuming that IDUC is effective for all data operation modes, assuming it is successful, if it is coded, it is a very strict coding mode of data DNA. I found it! I began to refine my sorting algorithm, word segmentation algorithm, ETL, YangLiaoJing, etc., and found one thing in common. All my works were refined to the rational function level that I could understand, which were small fragments of the combined decision-making process of adding, deleting, modifying and checking linear, multidimensional, database and memory data. These fragments can be coded effectively.

IV III DETA C Finding initions application

With this in mind, I have determined a ternary mapping coding mode of DNA to ETL neuron nodes, AOPM -VPCS- IDUC 3D coding mode. 4*4*4 Then each primitive is a 64-bit space, which is the computing primitive I have been looking for decades.

V DETA DNA decoding

If the AOPM -VPCS- IDUC 3D computational neuron mapped by DNA IDUC is established, how to decode it? I thought about what I have, about YangLiaoJing! It is the only way that I can do at present to construct the system of YangLiaoJing and demonstrate this idea and technology. It is still the absolute focus of that sentence.

V I DETA DNA decoding history

At present, what we know is that DNA peptide group has billions of long, double chains and 24 pairs of chromosomes. there are five primitives of ACGTu. if acgt can encode human higher intelligence logic, then humanoid data DNA with IDUC unit can also write hundreds of thousands of business transaction processing logic of AOPM VPCS. these two logics do not conflict.

V II DETA DNA decoding development

These two logics do not conflict. Are they one? I got an exciting argument, and I had to find an argument, so I screened my study and work style in recent 20 years, my own thinking style and the execution logic style of my software, hoping to find a negative theory to overturn it, but unfortunately I couldn't find it. So I followed up and re-examined my soft works, optimized them, and found that once optimized to the edge of rational function to irrational function, they were all linear. Small fragments of the combined decision-making process of adding, deleting, modifying and checking multidimensional, database and memory data. These fragments can be coded effectively. AOPM -VPCS- IDUC seems to explain all the answers I want.

V III DETA DNA decoding application

In order to overthrow my argument, I began to look for arguments everywhere to attack this argument. First, I found the topic of eternal life. According to AOPM -VPCS- IDUC, IDMC is true. Since it can be perfectly explained, I found the topic of infection in COVID-19, that is, DUOP is true, which is an exciting conclusion! I have been searching for answers to all the problems for decades from AOPM -VPCS- IDUC, so I started mapping and coding as follows:

I	INCREMENT/ADD	*D*	DECREMENT/FILTER	*U* UPDATE		*C/Q*	QUERY/CHECK
V	VITIONARY/FEEL	*P/E*	EXECUTE	*C* CONTROL		*S*	SET/STATIC
A	ANALYSIS	*O*	OPERATION	*P* PROCESS		*M*	MANAGEMENT

I	增加	D	删除	U	改变	C	查找
V	视觉	P	注册	C	控制	S	静态
A	分析	O	操作	P	处理	M	管理

Eternal life PDNS

Metabolism PDNS

永生的肽团可以筛选出

I 增加	**D** 删除	U 改变	C 查找
V 视觉	P 注册	*C* 控制	S 静态
A 分析	O 操作	P 处理	*M* 管理

人类特征新陈代谢相关的肽团可以筛选出

I 增加	**D** 删除	U 改变	C 查找
V 视觉	P 注册	C 控制	*S* 静态
A 分析	O 操作	P 处理	*M* 管理

新冠相关的肽团可以筛选出   COVID-19 PDNS

I 增加	**D** 删除	*U* 改变	C 查找
V 视觉	*P* 注册	C 控制	S 静态
A 分析	*O* 操作	P 处理	M 管理

Allergy PDNS

人类过敏反应 相关的肽团可以筛选出

I 增加	D 删除	U 改变	*C* 查找
V 视觉	P 注册	C 控制	S 静态
A 分析	O 操作	*P* 处理	M 管理

I got a clear DNA theorem:

The essence of <u>DNA is a combination indexing linklist of four meta-operations of adding, deleting modifying and Querying data.</u>

10-04-2020 DC

VI IDUC DNA and Its Applications

These are all the later stories. The application is too wide. First of all, my ETL began to expand in the three-dimensional direction to better serve medicine. Secondly, virus immunology and immortal virus exploration will never stop. Why ETL is used as the expansion point is inspired by my osgi paper on October 17, 2013. It is as follows

（*The Darwin's Theory of The Artificial Intelligence*）

In the latest knowledge engineering structure, the traditional expert system occupies a dominant position, but the world's demand system is in a changeable operating environment, so the data persistence theory is a goal to strive for. Artificial intelligence software, too, can't escape the disadvantages brought by natural updating. Where artificial intelligence will go, it will be planned naturally. Just like Darwin's theory of biological evolution, the new intelligent system standards are naturally selected by needs, which is the central idea I want to express. In the past 50 years, some classic software can't escape the choice of demand, and finally it turns yellow and dim. Of course, some enterprises rewrite and upgrade their products desperately. Because of the aging of core developers, new reformers can't master the original development ideas and theories. Finally, the quality of products suffers a huge impact and suffers heavy losses. A new software development theory needs to be confirmed, which is my thought. Software, too, needs an evolutionary system with self-artificial selection.

Through the recent construction, design and coding test of UNICORN AI software, I found that many computer theories created by fantasy have great differences in actual programming analysis. I used JAVA-based language, and I found that the inheritance of JAVA did not meet the language standard with evolutionary thought, but its methodology in this initial evolutionary standard test was far superior to C/C++. I didn't bring any troubles to my actual programming when I wrote JAVA program in C style, but JAVA still needs to be improved. For example, you abstract a parent class, and the variable function of your subclass still needs to be written in the way of "OBJECT parent class = (subclass) parent class" to make subclass operation. If grandchildren inherit subclasses, how can OBJECT get grandchildren? (I use the OBJECT subclass to inherit the parent class, and then the OBJECT subclass = (grandson) subclass. In this way, the grandchildren get the operation), but this is a big problem of dynamic memory structure allocation! The design is rather cumbersome. JAVA still stays at the level of primary language evolution, and does not have advanced evolutionary ideas. Secondly, if a subclass has more than one grandchild, only the subclass can operate, and the parent class cannot operate accordingly. This is also a criticism. Is it realized by adding OBJECT subclass = (grandson) subclass and OBJECT parent class = (subclass) parent class? This is even more complicated.

Through the above description, I have my own views, but I still chose JAVA, even though it is cumbersome, but there is no mistake, because it will be more cumbersome to implement in the underlying language. There are more traps. It is a natural choice for artificial intelligence to choose JAVA. JAVA and C# are both high-level languages, but JAVA's personality is born to deal with data, because JAVA was a WEB language in its early days, and WEB has unique advantages in dealing with data information, which is a real example of JAVA evolving into a data analysis language. C# has been improving itself in this problem, just like JAVA, even like JAVA, but there is no system to evaluate it. The WEB data engineers who applied JAVA in the early days will not transfer to C#, so the biggest advantage of C# is that it is only applied to controls on WINDOWS.

Through this description, it only proves that the greatest advantage of any language is only reflected in its creative theory and thought at the beginning of its birth. Therefore, JAVA and C# are not comparable at all. Because their original creative theory, system and ideological structure are different. If JAVA and C# really fail, finally, through the prediction of evolutionary thought, JAVA will go in the direction of graphics, big data analysis, WEB and C# should go in the direction of interface, control and WINDOWS device integration. The evolution of artificial intelligence software is mainly divided into update of parent class, variation and inheritance of subclass.

JAVA is perfect for dealing with subclass functions, and people who have used JAVA to develop large projects are quite experienced in dealing with interfaces and inheritance. But is there any variation in JAVA? It can be said that there is no, for example, when the parent class PUBLIC attribute 1 = 0; Subclasses can't have the PUBLIC attribute 1=1, which is a mutation failure problem. JAVA is flexible, but not as flexible as scripting language. Secondly, I want to say that the variation of JAVA is a variation with quotation marks. Its characteristic is that subclasses modify the functions of the parent class, and subclasses of JAVA can modify the processing procedure of functions with the same name of the parent class. However, you have to make the subclass and the parent class have the same function names. This is a JAVA default mechanism, which executes the same name of the parent class first, and then executes the same name of the subclass. Then return to the parent class, and then return to the procedure of. Therefore, the function with the same name can be modified in subclasses, thus ensuring parameter variation. In this way, the software is also very flexible and unique in the actual writing process. Finally, I have a deep experience through the expression of language evolution thought and program evolution thought mentioned above. Every language needs its needs if it is to be deeply rooted, and its functions should be selectively evolved in the needs. Otherwise, this is the biggest reason why languages have been eliminated. I don't like to see various languages emerge one after another in today's world. This is the biggest criticism that many languages have not evolved and can't reflect their needs. Secondly, languages need to be extended, and the appearance of API class libraries and some architecture systems of high-level languages is a good proof of extension. Finally, variation is similar to scripting language, which is flexible and convenient.
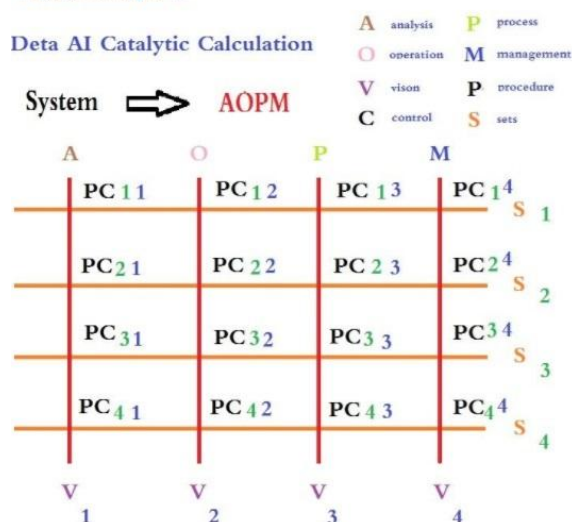
What about software? The same is true for software. It is particularly important to choose a language that suits your own needs. Secondly, the architecture of the software should have loose coupling, which is similar to OSGI and Felix. The OSGI idea of KNIME is consistent with that of LIFERAY. Although the API design style is different, the effect is very thick. Biology needs Darwin's thought, and artificial intelligence also exists, which is the basis of demand persistence. This is also the basic condition for my research and development of UNICORN AI platform.

Now I have enough confidence to continue to focus on the argument of making ETL mapped by my DNA code with evolutionary system reuse the perfect guarantee requirement persistence. On how to use ETL to map the code, I will go back to the previous year again and analyze the design idea of this picture at that time as follows

AOPM: https://lnkd.in/gmfRFGS
VPCS: https://lnkd.in/geMuGvC
SOURCE: https://lnkd.in/gySygmb
DEMO: https://lnkd.in/givi--n

Deta DNA Process

Deta AI Catalytic Calculation

System ⟹ AOPM

A analysis   P process
O operation M management
V vison   P procedure
C control   S sets

	A	O	P	M	
	$PC_1 1$	$PC_1 2$	$PC_1 3$	$PC_1 4$	S_1
	$PC_2 1$	$PC_2 2$	$PC_2 3$	$PC_2 4$	S_2
	$PC_3 1$	$PC_3 2$	$PC_3 3$	$PC_3 4$	S_3
	$PC_4 1$	$PC_4 2$	$PC_4 3$	$PC_4 4$	S_4
	V_1	V_2	V_3	V_4	

RST chromosome

DNA FATHER   DNA MOTHER

PDN FUNCTION	PDN MASK 1	PDN FUNCTION	PDN MASK 1
PDN FUNCTION	PDN MASK 2	PDN FUNCTION	PDN MASK 2
PDN FUNCTION	PDN MASK 3	PDN FUNCTION	PDN MASK 3
PDN FUNCTION	PDN MASK 4	PDN FUNCTION	PDN MASK 4

...   ...

$$\mathrm{E}\, PDN(I,J) * VPCS(I,J) = Humanoid\ DNA$$

love= pdn[4][4]{true,true,true,true,true,true,true,true,true,true,true,true,true,true,true}

study= pdn[4][4]{true,false,true,false,true,false,true,false,true,false,true,false,true,false,false}

function...= pdn[4][4]{true,false,true,false,true... ... ...false,true,false,true,false,true,false}

Yes, I have an idea already! ! Exciting, ETL node three-dimensional classification. this vocabulary

VII IDUC VPCS AOPM 3D Nero Cell and Its Applications

These are the following words. ETL begins to expand in the direction of 3D. First of all, I want to design the 3 D functional area of neurons based on the DNA mapping of DETA IDUC. This is the real humanoid independent thinking way that I can understand.
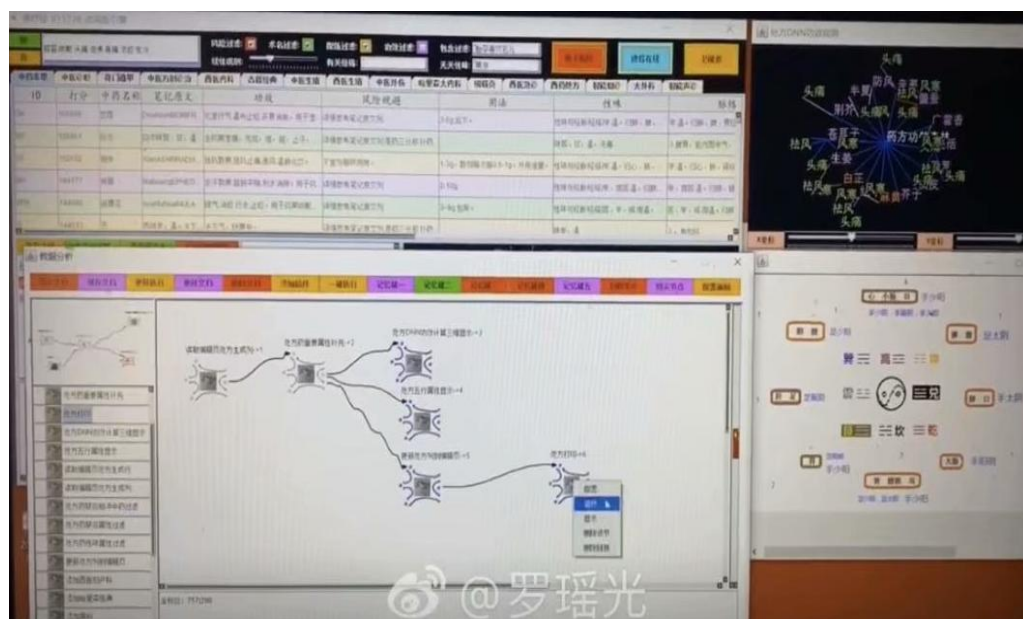
True Instance for YangLiaoJing
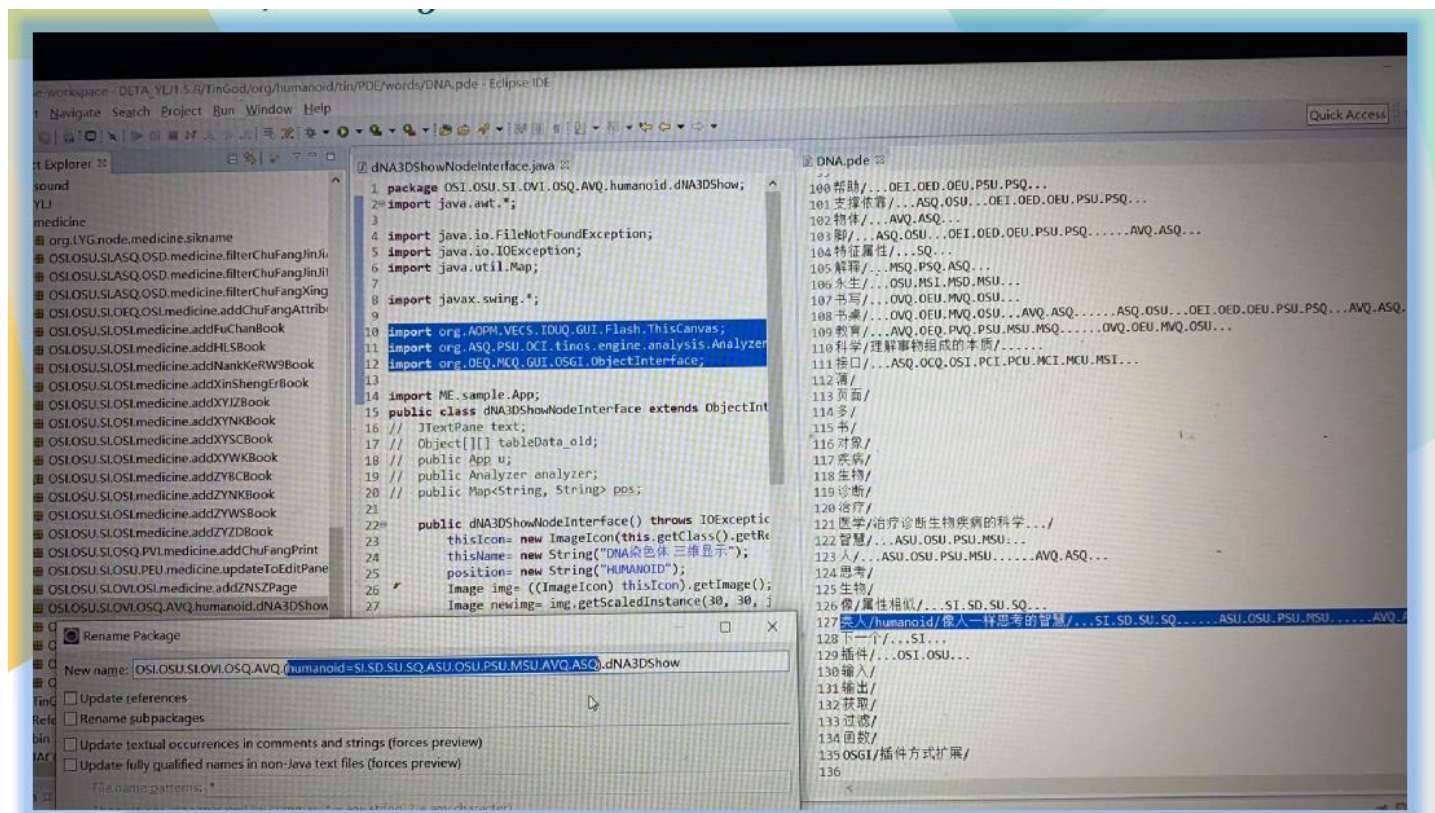
This is the first index application idea of DNA coding manual in human history.

I will let Org.lyg.node.medcine.addchufangattributeH.jar where change to Org.node.a.v.c.u.medcine.addchufangattributeH.jar

This a.v.c.u will form a dna mapping system code for analyzing visual control changes, which is convenient for future evolutionary optimization tests. After that, I will systematically encode these ETL index mapping sets into DNA index chains for YangLiaoJing. The ideas given to me in this paper are all creative ideas. Thank you for everything. I can first design A OPM V PCS IDUC initons 64-bit single chain for the integrity of coding, such as

AVI	AVD	AVU	AVC		API	APD	APU	APC		ACI	ACU	ACD	ACC		ASI	ASD	ASU	ASC
OVI	OVD	OVU	OVC		OPI	OPD	OPU	OPC		OCI	OCD	OCU	OCC		OSI	OSD	OSU	OSC
PVI	PVD	PVU	PVC		PPI	PPD	PPU	PPC		PCI	PCD	PCU	PCC		PSI	PSD	PSU	PSC
MVI	MVD	MVU	MVC		MPI	MPD	MPU	MPC		MCI	MCD	MCU	MCC		MSI	MSD	MSU	MSC

This index mode, even though it is not the final index of organic DNA of human beings, has become the first mapping execution mode of humanoid artificial design representing evolutionarily encoded DNA.

Not The End

In this paper, I spent 20 years of basic study, 7 years of work practice, 2 years of open source implementation, 18 corresponding data subprojects, 6 data fields, 2 big data works, 7 artificial intelligence papers, and gradually demonstrated some essence, such as

The essence of DNA is a combination indexing linklist of four meta-operations of adding, deleting, modifying and Querying data.

Byt the way:

Execution mode of neuron calculation: <u>a neuron time series calculation chain of specifical function calculation by mapping of DNA coding index reflection</u>

The essence of adapting to the environment is that <u>DNA coding indexes map related neurons compiler link to achieve better addition, deletion, modification and query the environment data.</u>

The essence of humanoid evolution offspring: <u>the offspring produced by optimizing the hybridization of the same coding logic part in the DNA encoding index chain mapped and retained by the neuron calculation method of data efficient processing</u>

It seems that the topic will never end, so I might as well boldly put forward new arguments, continuously and tenaciously focus and demonstrate, and I still enjoy it.

VIII Refer:

DETA PARSER, *https://github.com/yaoguangluo/DETA_Parser*
DETAETL, *https://github.com/yaoguangluo/ETL_Unicorn*
DETA Socket DB, *https://github.com/yaoguangluo/DETA_DataBase*
DETA TOP SORT, *https://github.com/yaoguangluo/sort*
SOFTWARE ENGINEERING *https://baike.sogou.com/kexue/d10131.htm?ch=fromsearch*
MVC, *https://baike.sogou.com/v25227.htm?fromTitle=MVC*
MVP, *https://baike.sogou.com/v70887934.htm?fromTitle=MVP%E6%A8%A1%E5%BC%8F*
ORACLE, *https://baike.sogou.com/v110535.htm?fromTitle=oracle*
VPCS, *https://github.com/yaoguangluo/DETA_Resource/blob/master/VPCS-Method_V1.1.doc*
AOPM, *https://github.com/yaoguangluo/DETA_Resource/blob/master/AOPM%20System%20On%20VPCS.doc*
YANGLIAOJING, *http://tinos.qicp.vip/DETA_HuaRuiJi.html*
HUARUIJI, *http://tinos.qicp.vip/DETA_HuaRuiJi.html*
SONAR, *https://www.sonarlint.org/*
DNA,
https://baike.baidu.com/item/%E8%84%B1%E6%B0%A7%E6%A0%B8%E7%B3%96%E6%A0%B8%E9%85%B8/78250?fromtitle=dna&fromid=98123&fr=aladdin
Darwin,
https://baike.baidu.com/item/%E6%9F%A5%E5%B0%94%E6%96%AF%C2%B7%E7%BD%97%E4%BC%AF%E7%89%B9%C2%B7%E8%BE%BE%E5%B0%94%E6%96%87/82699?fromtitle=%E8%BE%BE%E5%B0%94%E6%96%87&fromid=23890
NERO CELL, *https://baike.baidu.com/item/%E7%A5%9E%E7%BB%8F%E5%85%83/674777?fr=aladdin*
AI, *https://baike.baidu.com/item/%E4%BA%BA%E5%B7%A5%E6%99%BA%E8%83%BD/9180*
EN-DECODING, *https://baike.baidu.com/item/%E7%BC%96%E7%A0%81%E8%A7%A3%E7%A0%81*
PARAL COMPUTING, *https://baike.baidu.com/item/%E5%B9%B6%E8%A1%8C%E8%AE%A1%E7%AE%97*
HUMANOID, *https://baike.baidu.com/item/%E4%BB%BF%E7%94%9F%E4%BA%BA/5142593?fr=aladdin*
Data Mining, *https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%8C%96%E6%8E%98/216477*
UNICORN AI, *https://github.com/yaoguangluo/ETL_Unicorn*
KNIME,
https://www.baidu.com/link?url=g_8i8yfDrvNMqYfN6A9z_XoIc49s8yqzHgpYn-JCBIi&wd=&eqid=88da123b0000be28000000065f7eb5df
OSGI, *https://www.oschina.net/p/osgi?hmsr=aladdin1e1*
DML,

https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%93%8D%E7%BA%B5%E8%AF%AD%E8%A8%80/1082646 7?fromtitle=DML&fromid=10035808&fr=aladdin

John von neumann,

https://baike.baidu.com/item/%E7%BA%A6%E7%BF%B0%C2%B7%E5%86%AF%C2%B7%E8%AF%A4%E4%BE%9D

FFT KERNEL https://baike.baidu.com/item/%E8%9D%B6%E5%BD%A2%E8%BF%90%E7%AE%97/4756906

THREAD, https://baike.baidu.com/item/%E5%A4%9A%E7%BA%BF%E7%A8%8B

BLUE TOOTH, https://baike.baidu.com/item/%E8%93%9D%E7%89%99

PDN LINK, https://baike.baidu.com/item/%E8%82%BD%E9%93%BE/8625112?fr=aladdin

SDLC: "", https://en.wikipedia.org/wiki/Systems_development_life_cycle

VPCS: Yaoguang. Luo, https://github.com/yaoguangluo/VPCS_Theroy/blob/master/VPCS-Method_V1.1.pdf

INDEX, https://baike.sogou.com/v65431335.htm?fromTitle=%E7%B4%A2%E5%BC%95

SLANG, https://baike.sogou.com/v18863.htm?fromTitle=%E6%88%90%E8%AF%AD

SCRIPT LANGUAGE,

https://baike.sogou.com/v230334.htm?fromTitle=%E8%84%9A%E6%9C%AC%E8%AF%AD%E8%A8%80

PROSCRIPTION, https://baike.sogou.com/v5065331.htm?fromTitle=%E5%A4%84%E6%96%B9

XUESEN QIAN, https://baike.sogou.com/v237588.htm?fromTitle=%E9%92%B1%E5%AD%A6%E6%A3%AE

DIFFERENTAL, https://baike.sogou.com/v1966327.htm?fromTitle=%E5%BE%AE%E5%88%86

NEWTON, https://baike.sogou.com/v88517.htm?fromTitle=%E7%89%9B%E9%A1%BF

Leibniz, https://baike.sogou.com/v10692238.htm?fromTitle=%E5%B8%83%E8%8E%B1%E5%B0%BC%E5%85%B9

RECURATION, https://baike.sogou.com/v3195520.htm?fromTitle=%E8%BF%AD%E4%BB%A3

POLYNOMIAL, https://baike.sogou.com/v445974.htm?fromTitle=%E5%A4%9A%E9%A1%B9%E5%BC%8F

OSS Book: Eric Steven Raymond, 《The Cathedral and the Bazaar》

DETA parser source from GITEE: https://gitee.com/DETAChina/DETAParser

DETA parser source from GITHUB: https://github.com/yaoguangluo/DETA_Parser

DETA parser split method record: https://github.com/yaoguangluo/DETA_Parser/issues/21

DETA parser official demo: http://tinos.qicp.vip/data.html

DETA parser integrated products demo from GITHUB: https://github.com/yaoguangluo/DETA_Medicine

DETA parser integrated products demo from GITEE: https://gitee.com/DETAChina/DETA_Medicine

DETA parser integrated products demo download link: http://tinos.qicp.vip/download/HuaRuiJiTm1.0.3.zip

Quicksort YAOGUANG.LUO 4D source from GITHUB :

https://github.com/yaoguangluo/Data_Processor/blob/master/DP/sortProcessor/Quick_Luoyaoguang_4D.java

Quicksort YAOGUANG.LUO 4D source from GITEE:

https://gitee.com/DETAChina/DETA_Data_Processor_Pub/blob/master/DP/sortProcessor/Quick_Luoyaoguang_4D.java

Reflection on YAOGUANG's Peak Array Split Defect1.0

https://github.com/yaoguangluo/DETA_Resource/blob/master/Reflection%20on%20Yaoguang's%20Peak%20Array%20Split% 20Defect1.0.pdf

https://github.com/yaoguangluo/DETA_DataBase , this project is building based on VPCS 1.0 theory. From this project, sleeper is more likes a socket thread, Hall is more like a sleepers pool, the vision more like a http call, and dream are more like a database system management. For any question please check the reference links as bellows:

https://github.com/yaoguangluo/NeroParser , this project demonstrated the Chinese and English words parser for the NLP AI domain, it mostly uses VPC to build the algorithms engine and kept 27, 000, 000 mixed words parsers per each second.

https://github.com/yaoguangluo/DETA_VPCS_Frontend, this project is a high sufficiency and tiny HTTP socket web, it only contains java script and java source file. Boot takes 9ms on the VPCS engine.

https://github.com/yaoguangluo/DETA_CACHE, this project is a rest calls cache by using VPCS TCP method. It seems very tiny and fast.

https://github.com/yaoguangluo/DETA_BackEnd, this projects is a backend rest service by using VPCS.

https://github.com/yaoguangluo/ETL_Unicorn,this, project is a data mining ETL software by using VPC, similar with the KNIME. Author makes high recommendation about this portrait.

Thanks GIT Hub, Eclipse.org, Microsoft, Apache and Java. Google, Angular, JQUERY, Bootstrap, JSON, Oracle.
Thanks from the bottom of my deep heart.
Thanks to my family, my colleagues, and my customers. Common!