The INITONS Catalytic Reflection Between Humanoid DNA and Nero Cell

YAOGUANG LUO, RONGWU LUO
Liu Yang DETA Software Development Limited Company, 410300, China
Changsha Le Hao Fu Chan Hospital,410300, China
E-mail: 313699483@qq.com

Abstract: VPCS[9] architecture is not the end. Absolutely, At least at this paper, will make an implementation in five section: DETA HUMANOID[20] cognition, DETA Medical Business backend logic, DETA Catalytic computing, DETA Finding INITONS, DETA DNA[14] DECODING[18]. Above all There will spend more and more words in my DETA DNA[14] Law of IDUC. And its applications in the real world.

Keywords: VPCS[9] AOPM[10], IDUC, Nero, Artificial, Decoder, Medical, PARAL[19]ling, Computing, Humanoid, ETL, Parser[1], Data Mining[21]

1 DETA HUMANOID[20] cognition

Since the INITON of the DETA OSS, there has a lot of questions where based on the HUMANOID[20] DNA[14] catalytic computing, for example AI[17], Plan a starts as below. Code a problems solution software like a way of YAOGUANG Luo 's cog-style life.

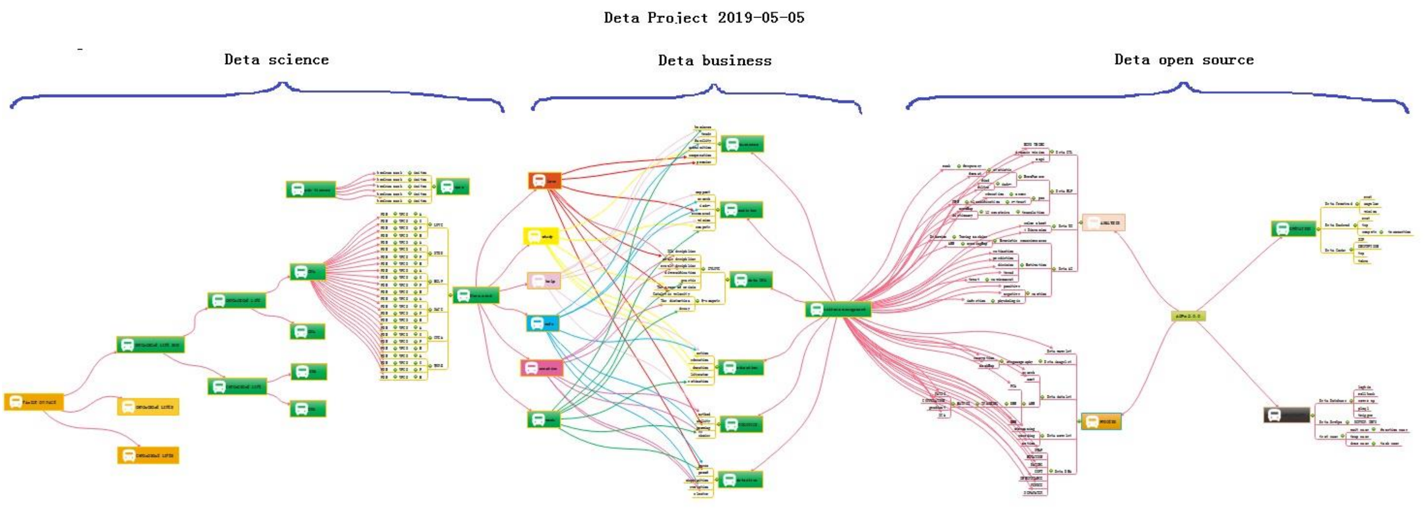


Figure 1 DETA OSS NUWA PLAN

Look at this Figure 1, smartly, build basic foundations first, then create more business software where based on this foundations, and finally swap to a HUMANOID[20] model. Many times, hope the model could be an Immortality

1.1 DETA HUMANOID[20] 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[11], 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[11] system.

1.2 DETA HUMANOID[20] 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[15]'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[12] 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.

1.3 DETA HUMANOID[20] cognition application

There are many applications of HUMANOID[20] 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,(过滤当前政治,经济实体名,人物姓名.), 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, (过滤当前政治,经济实体名,人物姓名.), (过滤当前政治,经济实体名,人物姓名.), (过滤当前政治,经济实体名,人物姓名.), 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[5] 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[2] of DETA is ready to go in this direction. The author designed a paper last year to describe the application mechanism of AOPM[10] as follows: AOPM[10] Open Source System On SDLC[31] Theory

Introductions, Recently my colleagues take more care on the SDLC[31] evolution of open source software engineering[5], 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[31] what be my main task now. after imagination and logic recursion, the key is an optimization of ordinary SDLC[31] such as Water Fall Model. First time for makes an introduction to waterfall of SDLC[31]? 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 below.

Topic: Ten Definition of The Open Source, OSS Book[42] Reading Note, 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 news 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.

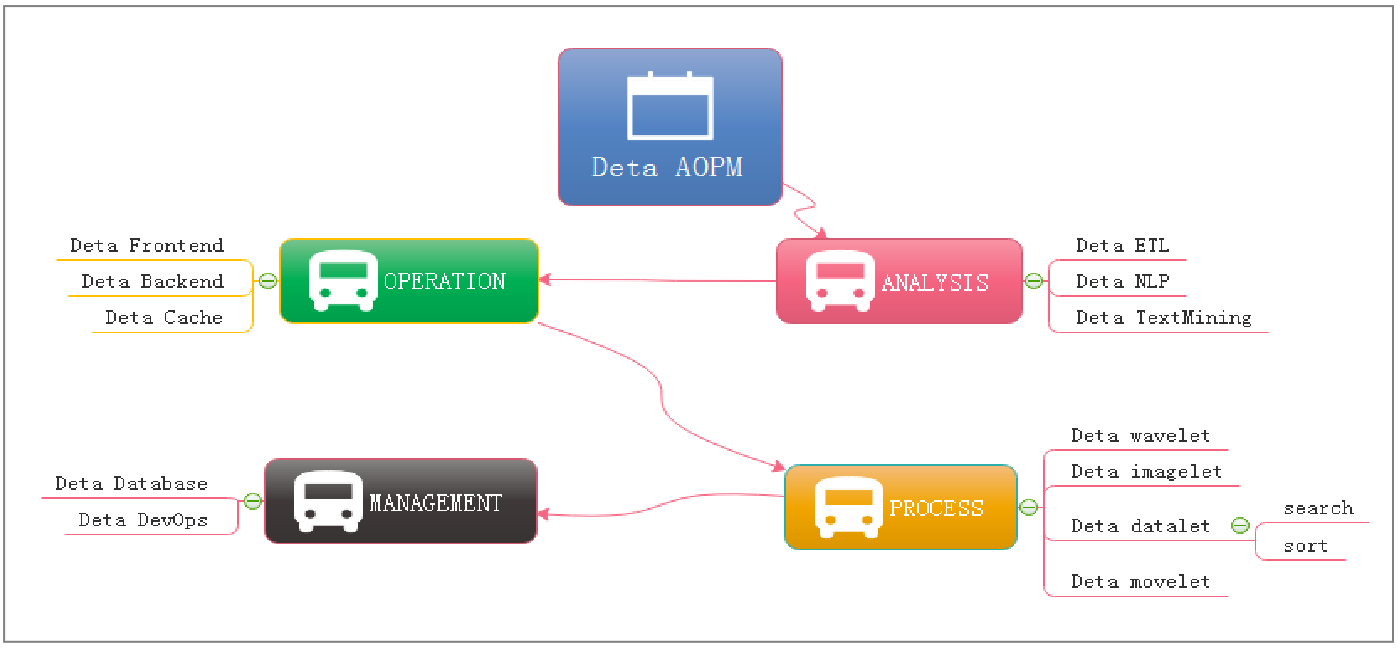


Figure 1-1 AOPM[10] Applications with SDLC[31]

Evolutions Last year almost the same question: how have you done so many projects during the year of 2018-19? absolutely: connection. Always, with connection, 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[2], they both have the same attributes such as AI[17], Analysis and Data etc. With this connection, my tasks became more dynamically. 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[17], now is a Bazaar requirement, but 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[47] system. This article describes the LINUX[48] development with the revolutionary road, as the process from the bazaar to the cathedral. First, the author tells the contrast between UNIX[47] and LINUX[48]: now UNIX[47] 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[48] 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[47] 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[47] 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[47] feels like a cathedral. The birth of the LINUX[48] 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[48] development meets the needs of all groups. Similar a huge bazaar. The author commenced a Lenovo, a conclusion that LINUX[48] will eventually beat UNIX[47], UNIX[47] gets the range of fresh blood is less than the LINUX[48]' s, also the number of the UNIX[47] team members is less than the LINUX[48]' s. UNIX[47] customers and employees are aging. But LINUX[48] development more in line with the user of the needs. Its own development is to establish a relationship on this demand and requirement. LINUX[48] is young now. Summary, UNIX[47] and LINUX[48] 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[48] because it is better adapted to the environment.

At figure 1-2, DETA open source main based on AI[17] domain, it already formed as an ecology system, go ahead to the application, thanks.

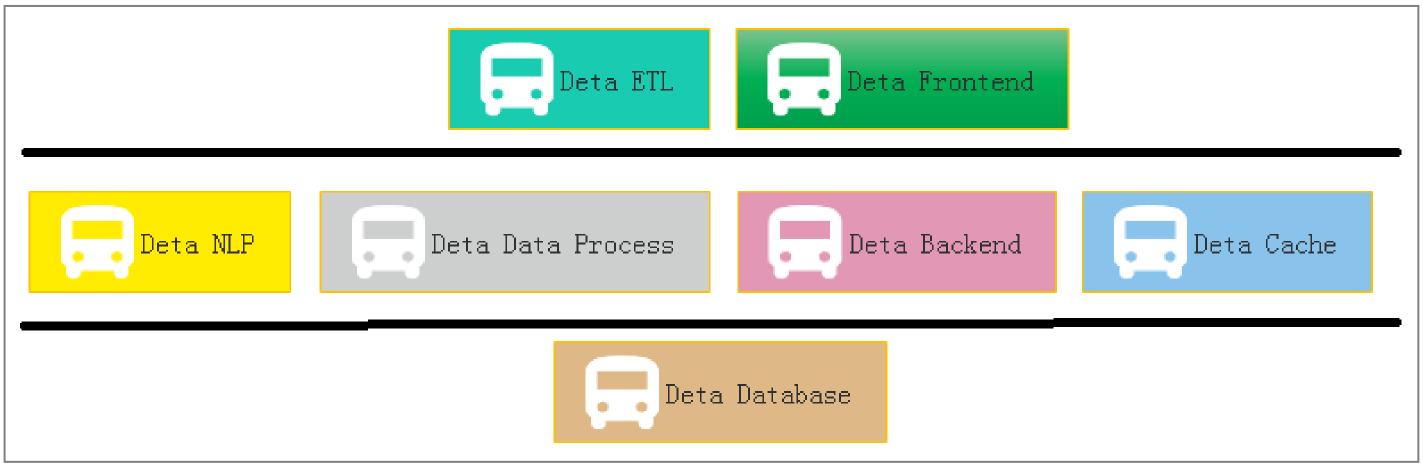


Figure 1-2 Sections of DETA Projects Group

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.

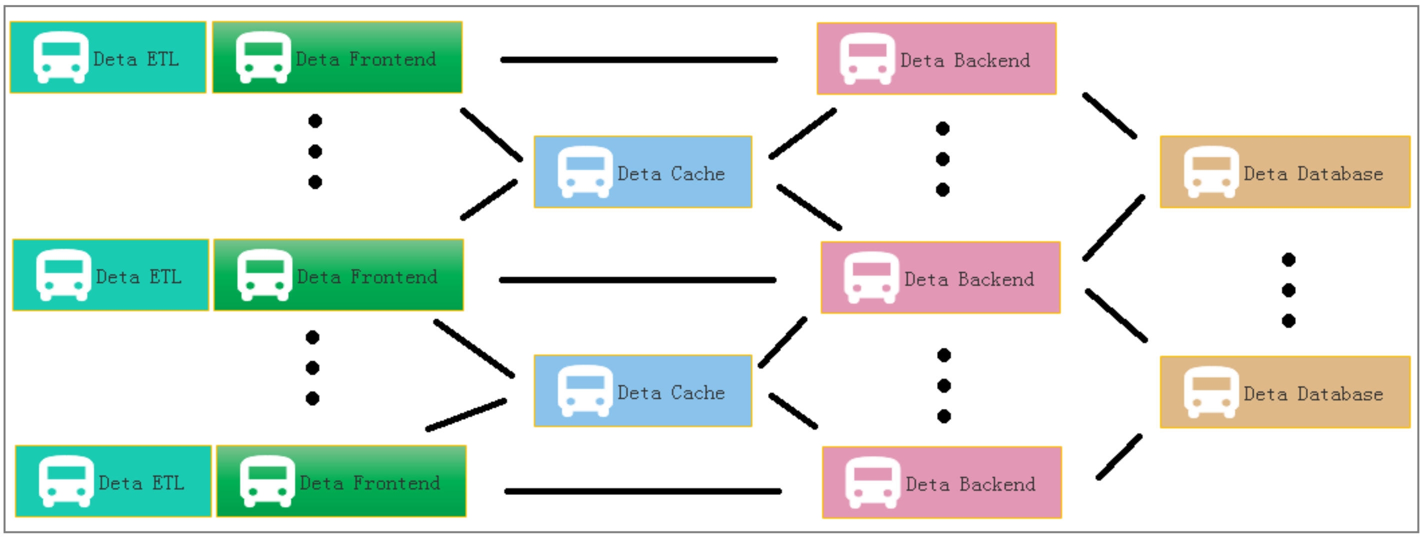


Figure 1-3 DETA WEB Projects System

At Figure 1-4 is a real sample for web DEVOPS by using DETA Open Source.

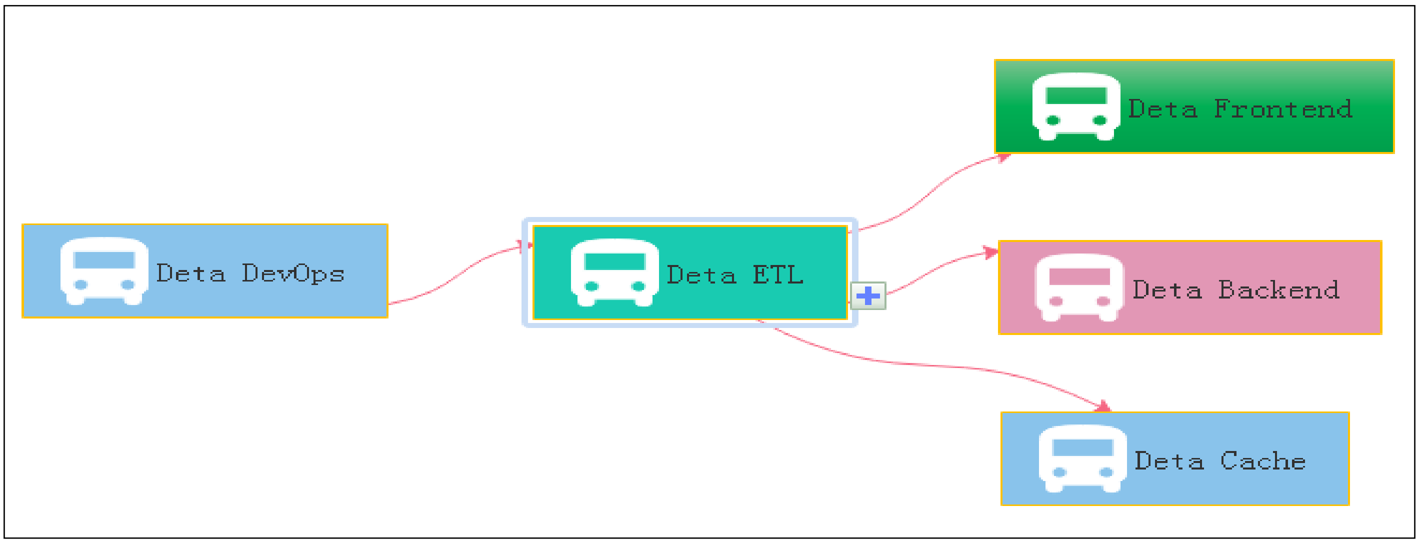


Figure 1-4 DETA DevOps Projects System

2 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[20] wisdom to meet the rapid development of business intelligence applications.

2.1 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[6] architecture in Shanghai FANTENG Information Technology Company. At that time, the feeling was that MVC[6] could solve all kinds of business logic. In the same year, the author first came into contact with MVP[7] to do multi-thread Bluetooth big file project, and felt that MVP[7] 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[6] architectures, such as Spring, Martini, etc. The author thinks that gives MVC[6] an intelligence urgently.

2.2 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[12] Medical Big Data System. At that time, spring boot, MYSQL were too heavy. If the database rest handshake system of Socket[3] stream was designed according to CGI, many problems could be solved easily. So 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. 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[9] structure shoots me at present.

With regard to the excessive description of VPCS[9], 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[15]'s theory of evolution, an effective VPC programming concept is elaborated based on the neutral coupling of MVC[6]+MVP[7]. 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.

2.3 DETA Business backend logic application

In the whole year of 2019, the VPCS[9] 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[9] is gradually integrated into the works of YANGLIAOJING[11] and HUARUIJI[12]. Of course, there are many shortcomings, the biggest one is that they do not repair themselves. Although I designed the sleeper and hall keeper mechanisms, these mechanisms are only the corresponding business logic units that I complete through decision trees, not HUMANOID[20] evolutionary thinking. At least, I don't think they are HUMANOID[20] intelligence. To be precise, at present, they are only artificial intelligence, a kind of artificial intelligence logic corresponding to AOPM[10] and VPCS[9], but not the HUMANOID[20] evolutionary intelligence logic that I want.

Started to explore HUMANOID[20] computing again. About the application principle description of VPCS[9], the author designed a paper as below: VPCS[9] Backend Theory And Its Application.

Introductions: Let see the verbal keys, 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 2-1.

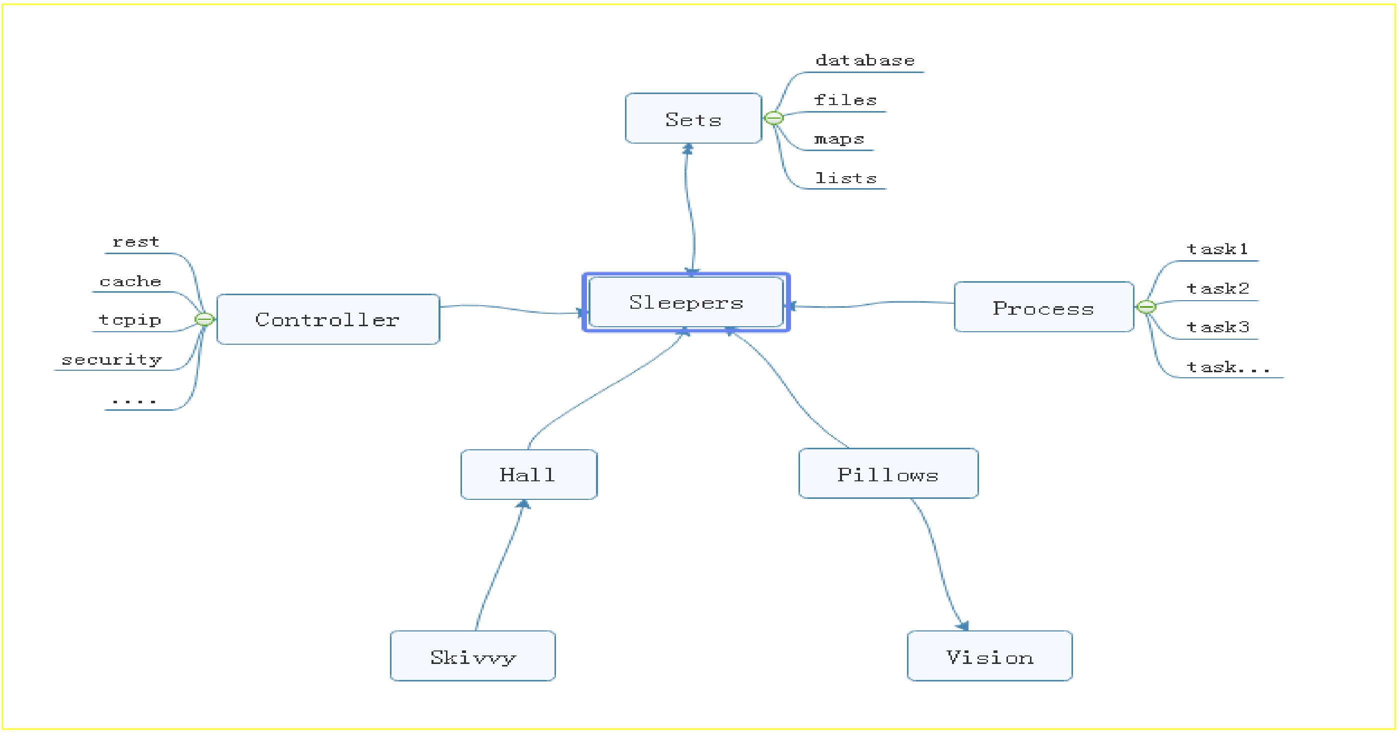


Figure 2-1 VPCS STAR MODEL

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. 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….".

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 makes an explanation for VPCS[9]. The VPCS[9] 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[5] 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 they want to build the web service to get rest call, and return the JSON feedbacks, 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[3], 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 2-2.

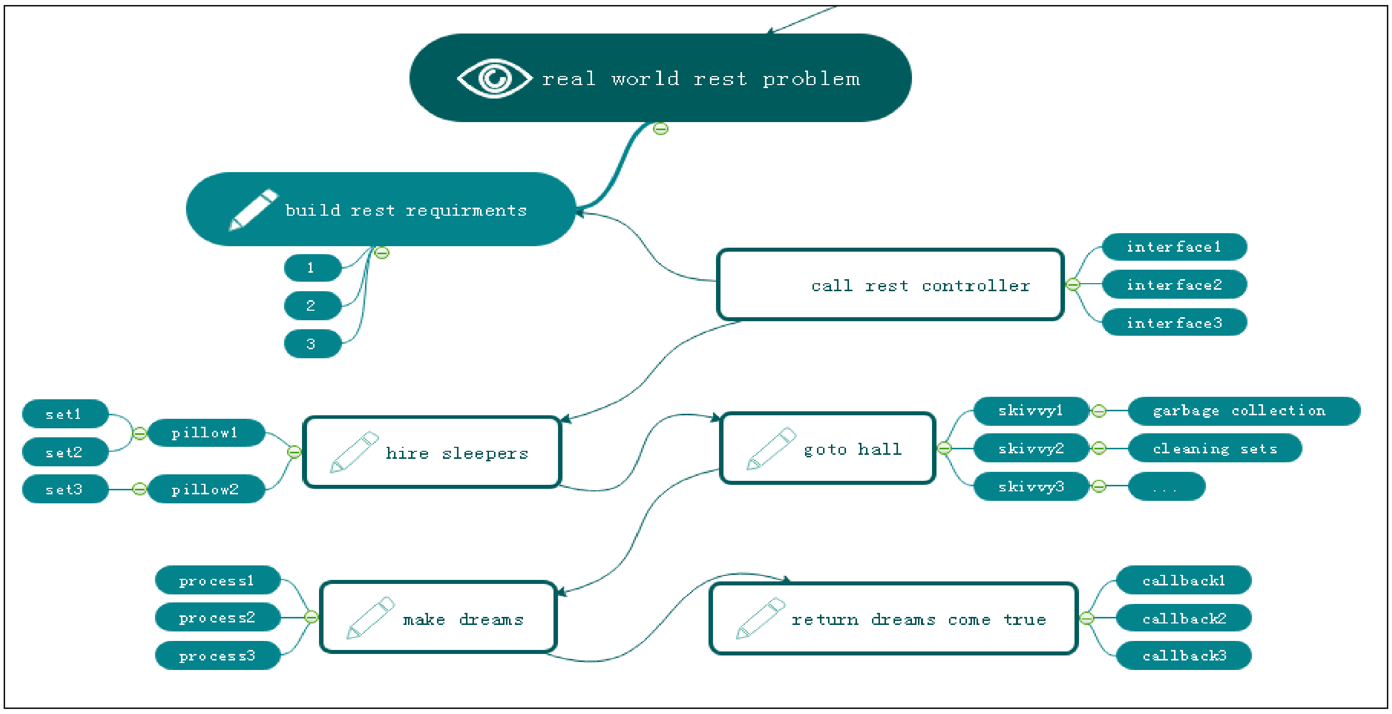


Figure 2-2 VPCS BACKEND MODEL

Focus on this landscape, mostly different to the MVC[6]: Model View Controller, MVP[7]: Model View Presenter or other architectures 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[9], desktop App once said: VPCS[9] 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 2-3.

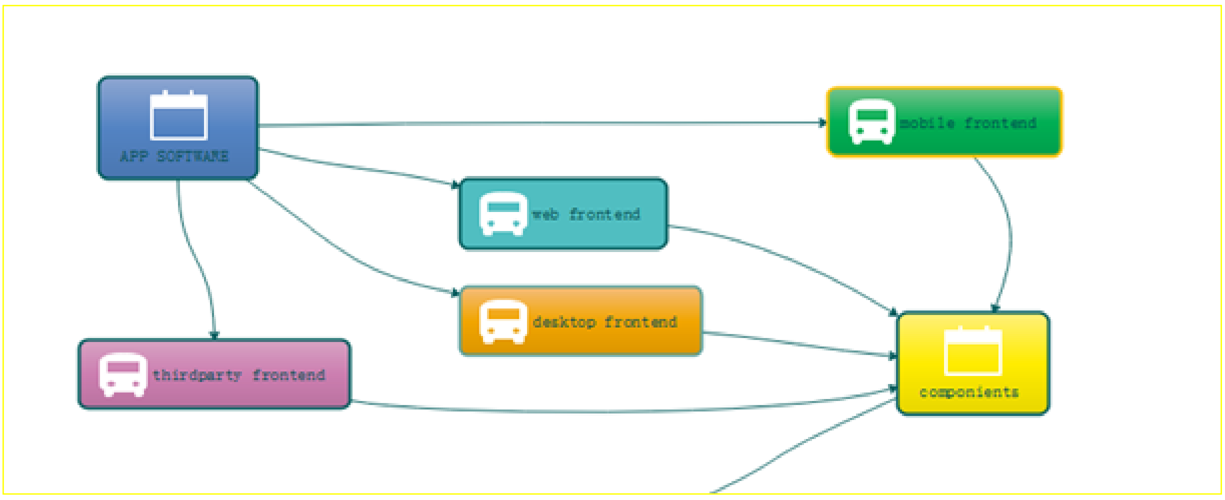


Figure 2-3 VPCS WORK WITH FRONTEND

From this picture, all of the software can be fast and safe while using VPCS[9], because it is already separated out the big system into backend and frontend two parts. and VPCS[9] keeps safe and fast in the backend section. Compare to the MVC[6], VPCS[9] will get more cautious and Details, and compare to MVP[7], VPCS[9] also will get more safe and high efficiency. In the common software engineering[5] 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[9] system, don't care about what they do for the front end, 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[9] will say: concurrent functions are safe functions. We guess in the future REST-VPCS[9] will be used in multiple WEB service. Especially in the high speed, efficiency, micro web systems with high level security for example medicine, DNA[14], cloud server, electronic police system and ecommerce systems etc.

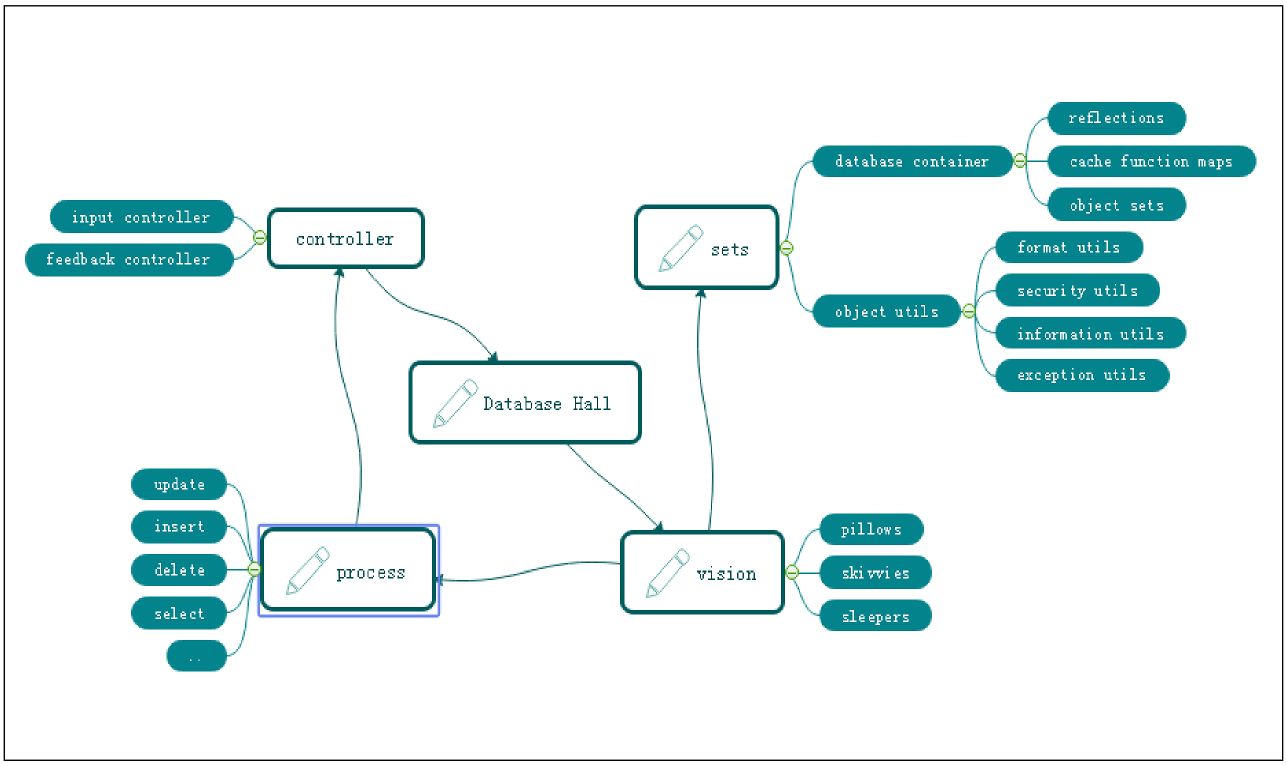


Figure 2-4 VPCS FOR DATABASE SYSTEM

As the Figure 2-4, a new method of the Database system designing shows us VPCS[9] is a pretty way for the modern data information management system. Definitely used in the DETA Database[47] system. For this instance, 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[9]. 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 2-5 the hall building need a single instance like a home keeper but here is a hall keeper, any else, this person is very important for keeping the VPCS[9] 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[9] system. Call the hall keeper. will know everything about skivvy's work status.

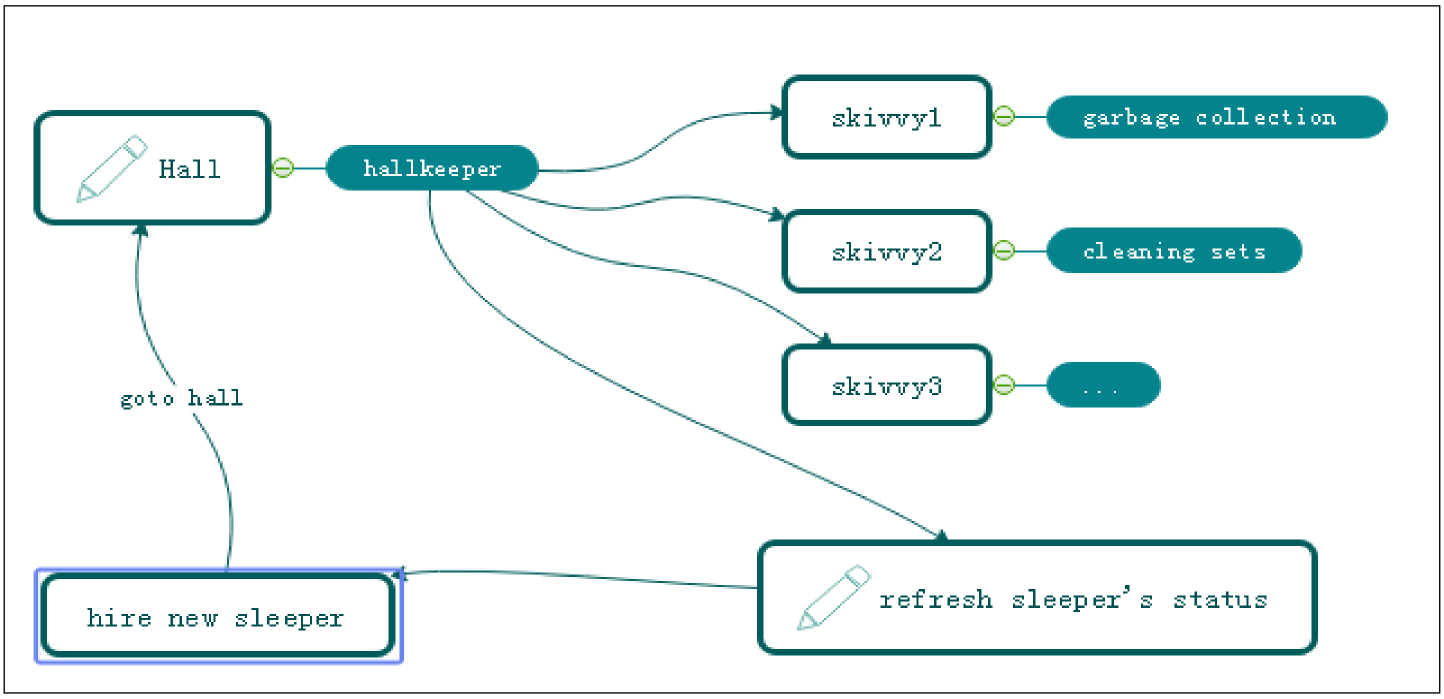


Figure 2-5 VPCS KERNEL

How does sleeper doing? Make a dream? Cool, in the VPCS[9] 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[9] 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[9], because they really want to know why VPCS[9] is faster and safe. Ok, please see the Figure 2-6, the answer is absolutely, VPCS[9] main components of the time sequence only contains five aspects. Almost similar like the hotel management. Certainly, 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[9] 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[9] only 30kb.

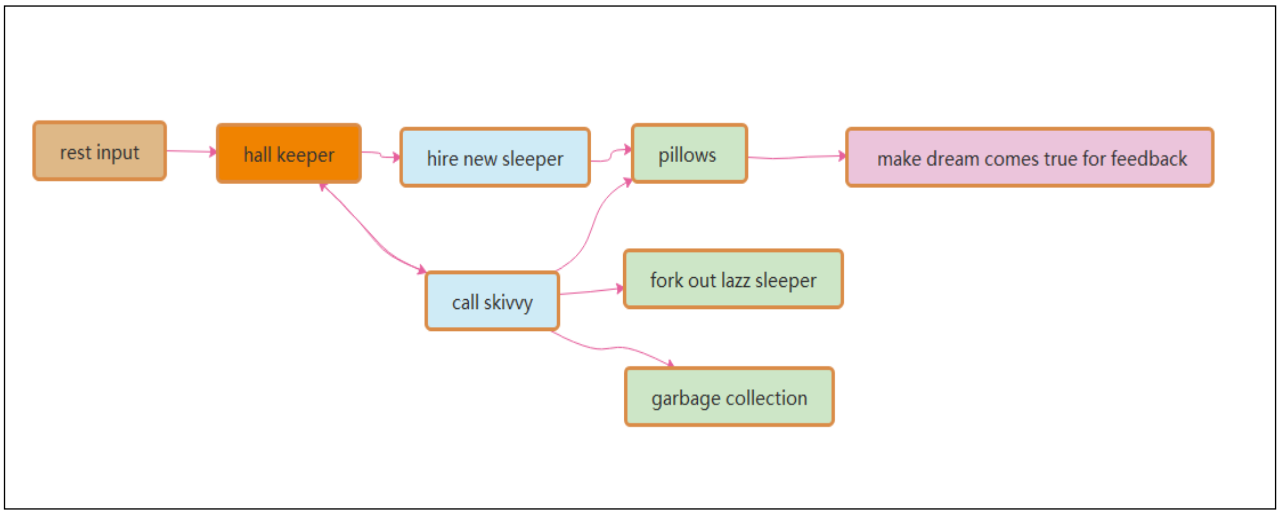


Figure 2-6 VPCS Sequence Diagram

How does the hall build? 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 2.6.1, this sample is a true demo in the real world for the WEB rest service. It's very important to create a player role such like hall keeper. what would like 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.

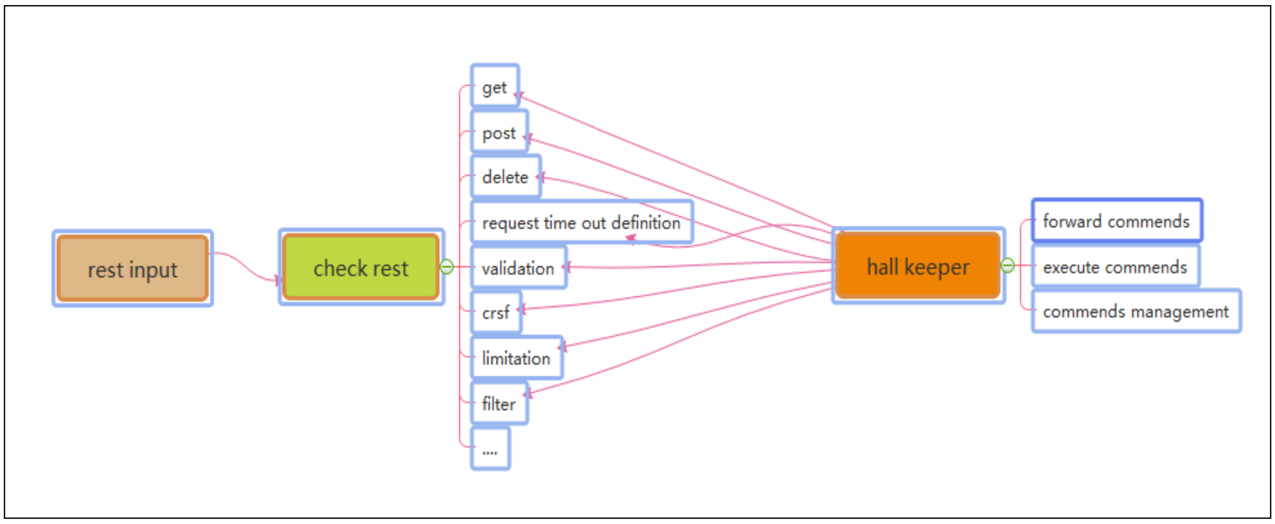


Figure 2-6-1 The Interaction Between Rest Call and Hall Keeper

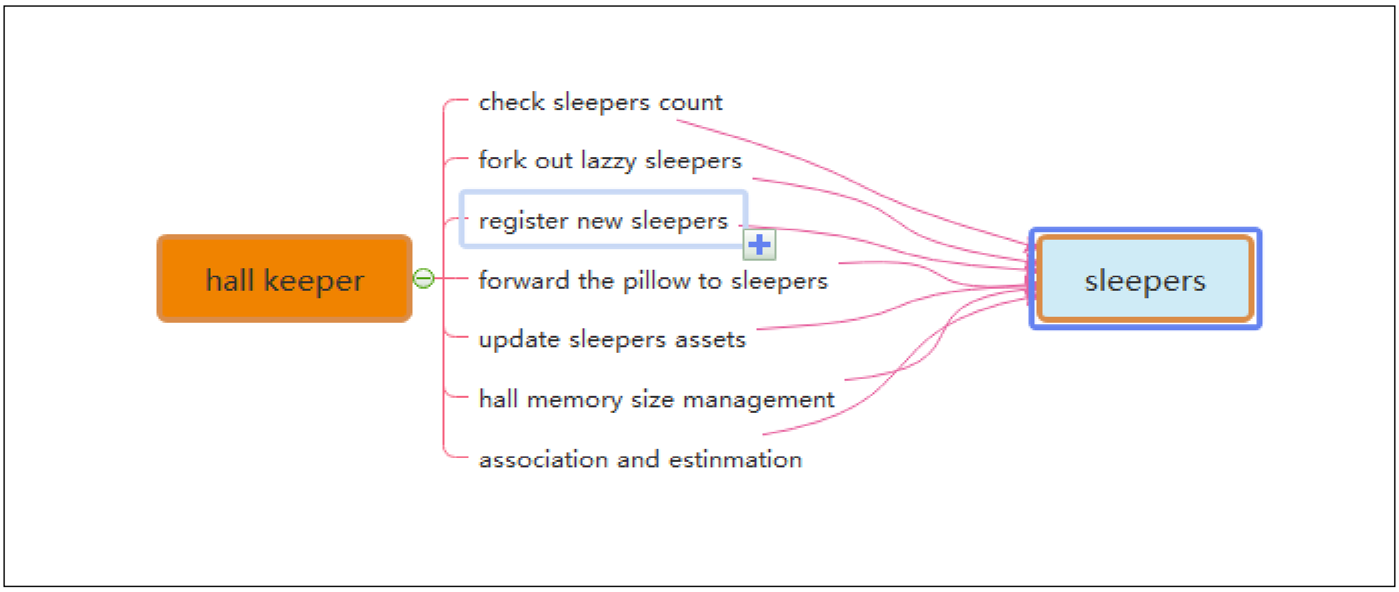


Figure 2-6-2 The Interaction Between Hall Keeper and Sleepers

Many of these software developer also asked how and why VPCS[9] 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 identify 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 2.6.2.1 VPCS**[9]** kernel.

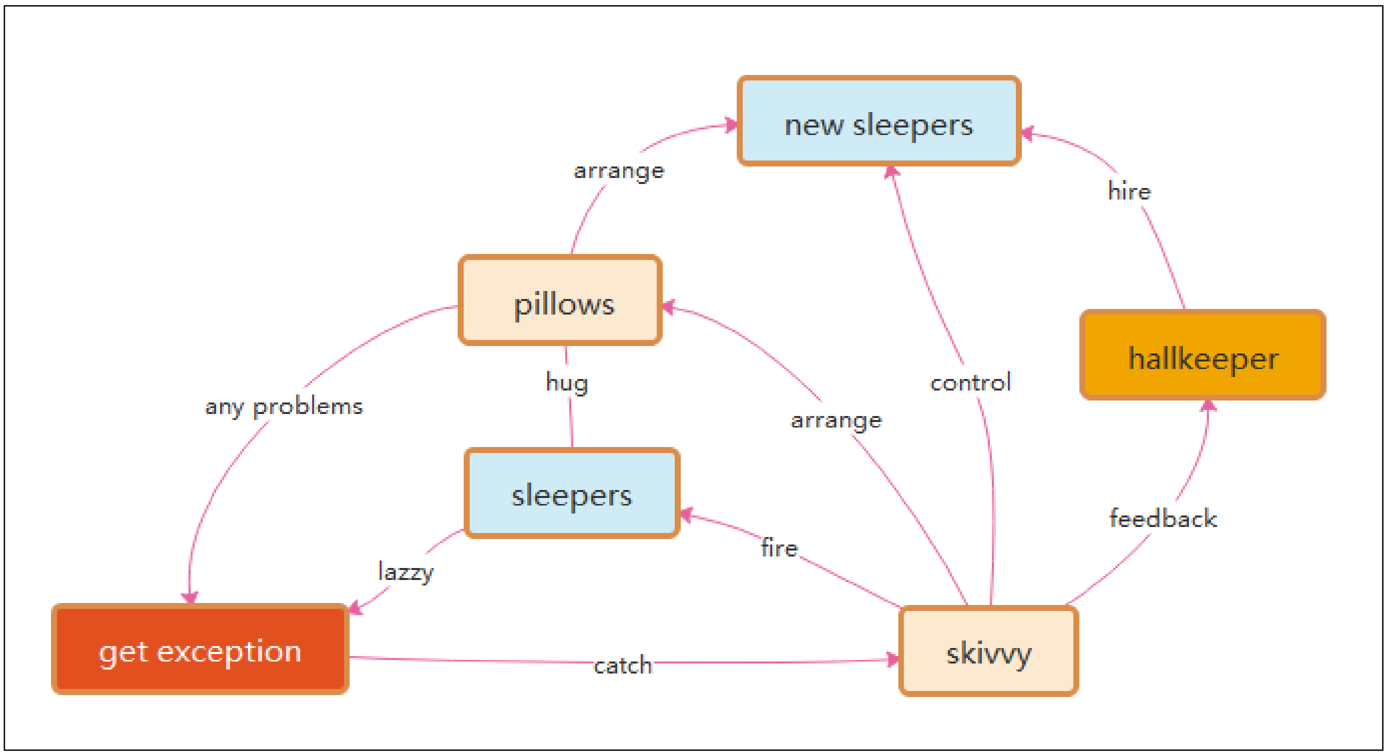


Figure 2-6-2-1 VPCS kernel

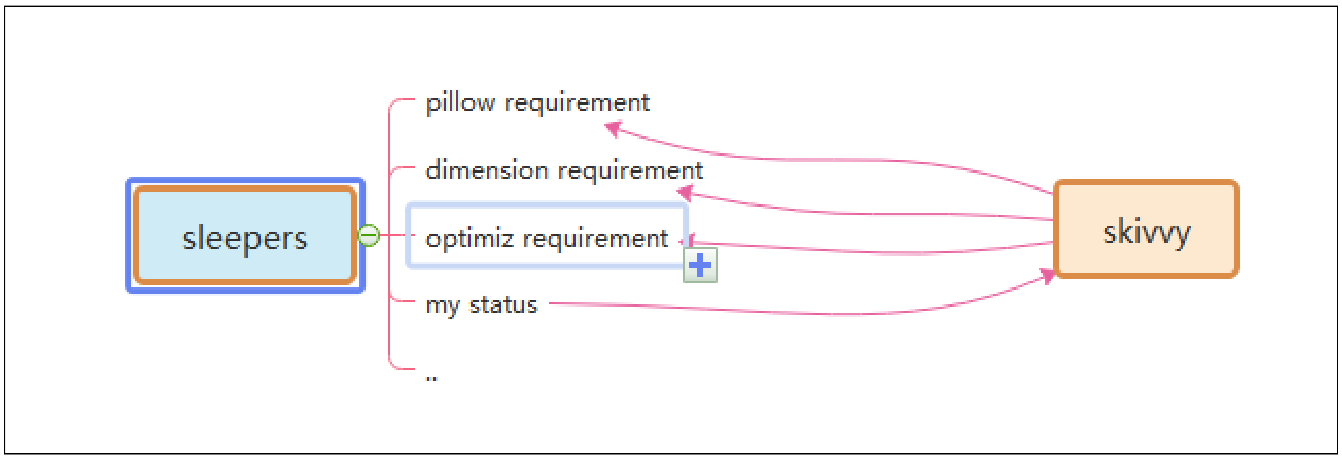


Figure 2-6-3 The Interaction Between Sleepers and skivvy

Many times, DEVOPS, they really worry about VPCS[9] 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[9] supports us for daily works? Because of Hall Keeper and skivvy, as Figure 2-6-3 and 2-6-4. 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 a true answer.

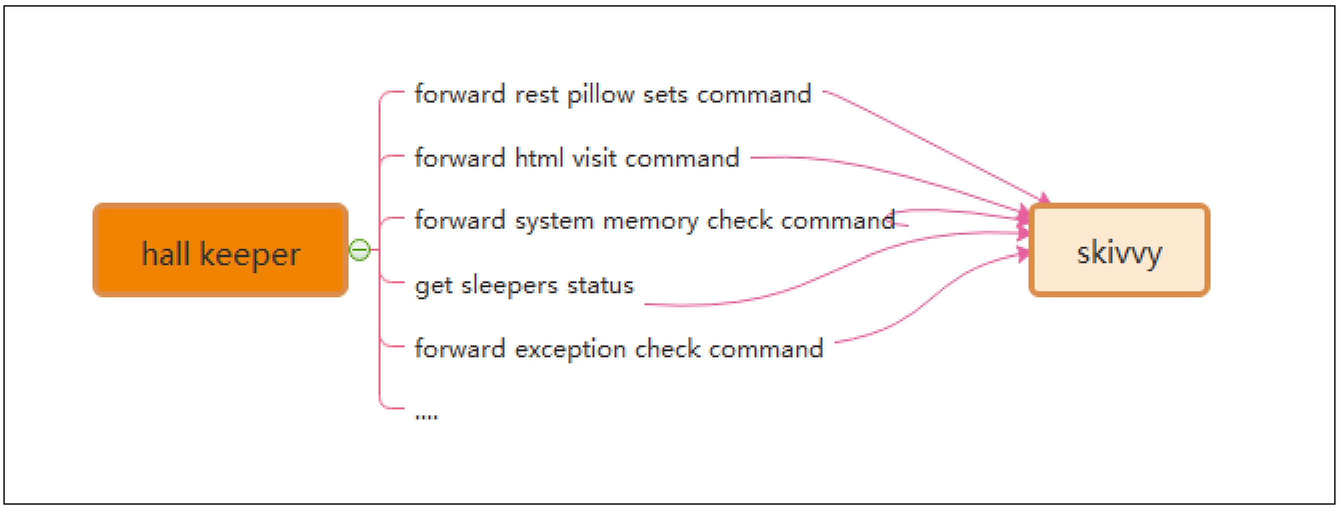

hall keeper
forward rest pillow sets command
forward html visit command
forward system memory check command
get sleepers status
forward exception check command
....
skivvy

Figure 2-6-4 The Interaction Between Skivvy and Hall Keeper

Recently Mr. Yang 1291244774@qq.com who asked me about VPCS[9] 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[9] seems so smart.

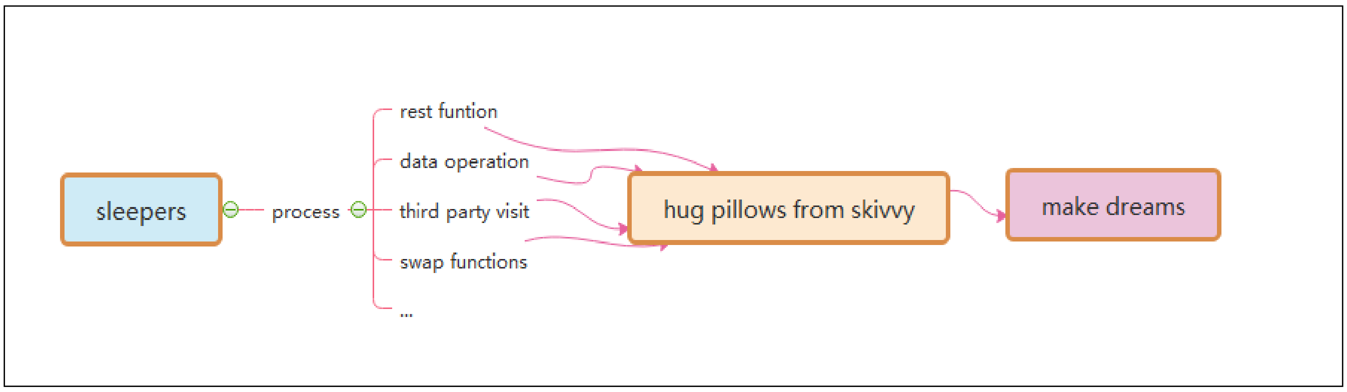


Figure 2-6-5 The Interaction Between Sleepers and Pillows

3 DETA Catalytic computing

How to realize human computing? How to start? Do basic research! According to the science notes and digging some unknown basic knowledge. Very happy, because 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[12], UNICORN ETL[2], Socket[3] 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.

3.1 DETA Catalytic computing history

The refinement method of DETA's first catalytic calculation is first reflected in DETA parser[1], 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.

3.2 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. But 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. 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.

Goal one: Quicksort YAOGUANG.LUO 4D

3.2.1 Details：

See Figure **3.2.1.1** For example the array input as below where gave 11 digits.

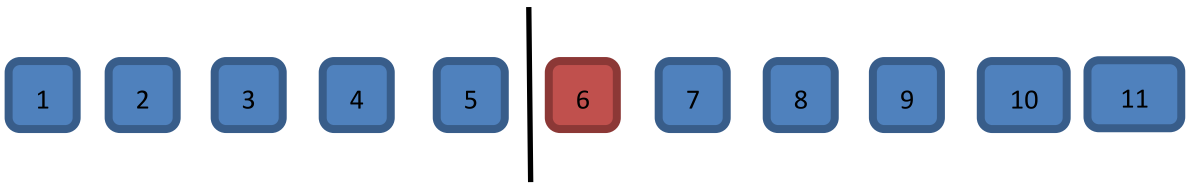


Figure **3.2.1.1**

See Figure **3.2.1.2** The first split, we could see the digit-6 will auto arranged to the right part.

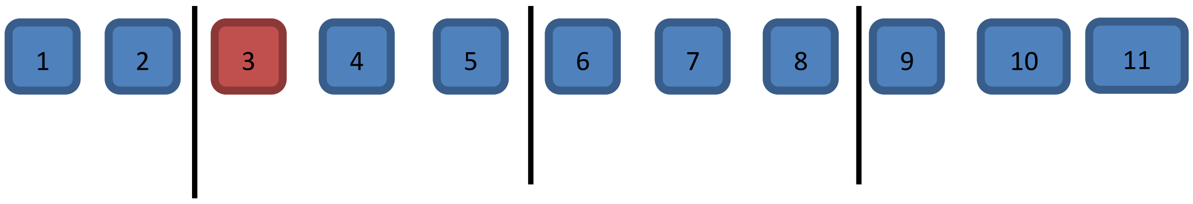


Figure **3.2.1.2**

See Figure **3.2.1.3** And the second split, may see the digit-3 will be auto arranged to the right part

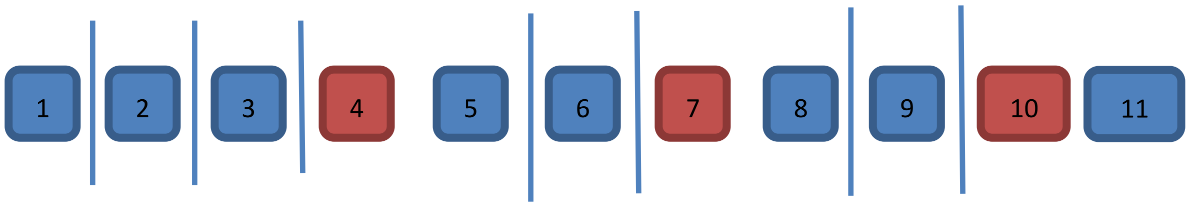


Figure **3.2.1.3**

See Figure **3.2.1.4** The third split, we may see the digit-4.7.10 will be auto arranged to the right part

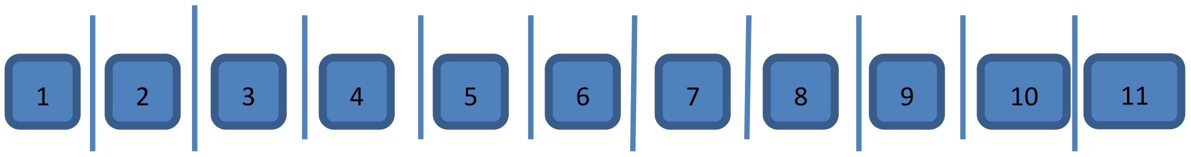


Figure **3.2.1.4**

3.2.2 Thinking:

After the split array showing, the big problem about the asymmetry defect, 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, 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.

3.2.3 Problems:

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

3.2.4 Solutions:

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, reduces the compute sets. For any less memory system, reduces more and more memory garbage after reduce the inner register or temp value sets. Third, makes an optimization of the function logic where to instead the old complex functions. Those ways include the condition, algorithm, method or discrete optimization. End, uses mathematics of double differential, deep definition, acquisition or POLYNOMIAL[41] to get the solutions.

See Figure 3.2.5 True Top Sort[4] 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){// reduce the condition differential check, reduce the recursion loops
                while(!(a[lp1++]> x|| lp1> rp)) {                    }
                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;
}
```

Figure 3.2.5

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：

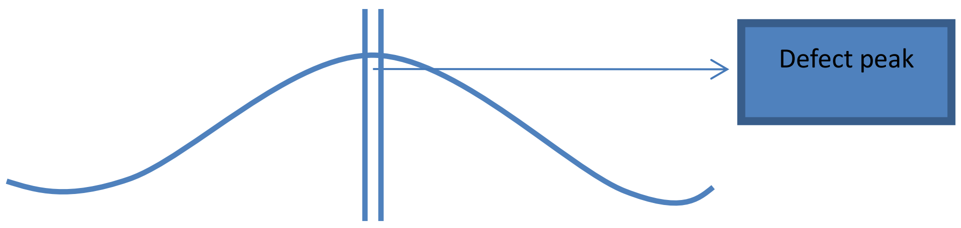


Figure 3.2.5.1

See Figure 3.2.5.1 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[1], 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 See Figure 3.2.5.2.

```
int x= a[lp]< a[rp]? a[lp]: a[rp];
```

Figure 3.2.5.2

Third, See Figure 3.2.5.3 De Morgan condition differential as the code below, it will cause the condition ability accelerations.

```
while(!(a[lp1]>x|| lp1>=rp)) {
```

Figure 3.2.5.3

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

Goal Two: DETA parser[1]

3.2.6 Details：

See Figure 3.2.6 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?

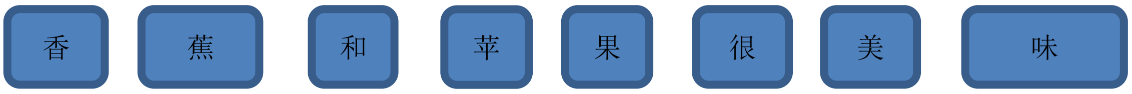


Figure 3.2.6

See Figure **3.2.6.1** convolution length indicate by marching Nero INDEX[32] tree as below: 2|1|2|3

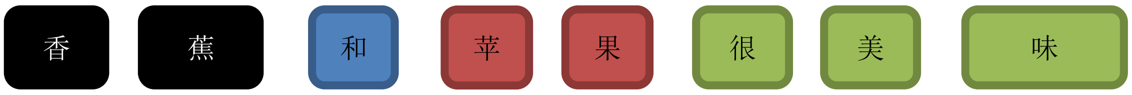


Figure 3.2.6.1

See Figure **3.2.6.2** convolution POS indicate as below: n |c |n |adv |adj

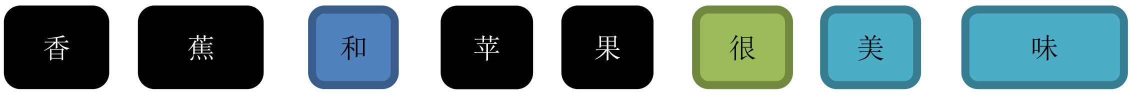


Figure 3.2.6.2

See Figure **3.2.6.3** convolution split

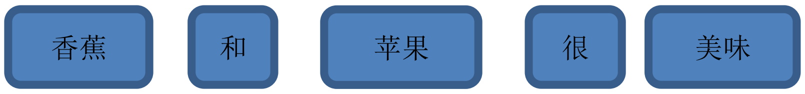


Figure 3.2.6.3

3.2.7 Thinks:

DETA project for the Chinese segment separations, the more and more important problem is the POS frequency peak waste, POS flow functions will spend a lot of time to do the low prior convolution split condition check first…

3.2.8 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.

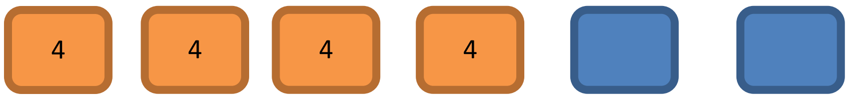
3.2.9 Solutions:

First, format the convolution kernel of the INDEX[32] dictionary tree, for example use the char ASCII as the INDEX[32] length to reduce the match time of the convolution length indication. Also, define the POS convolution kernel size less than five. Second, 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.

3.2.10 True instance:

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

See Figure **3.2.10.1** check 4 chars SLANG [33]



See Figure **3.2.10.1**

See Figure **3.2.10.2** Check 3 chars key word

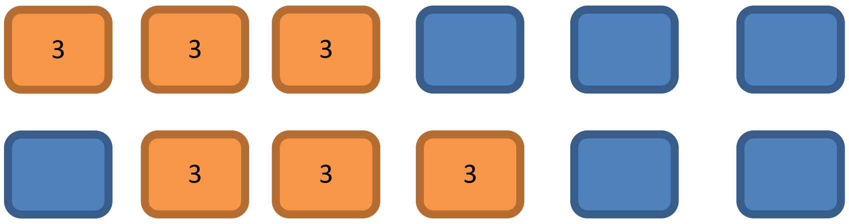


Figure **3.2.10.2**

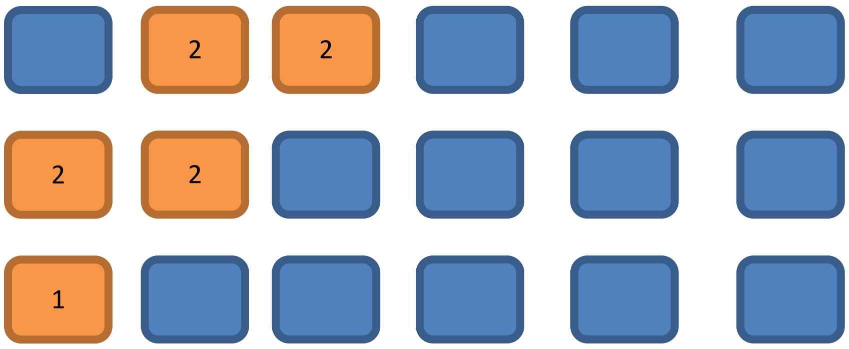


Figure **3.2.10.3**

3.2.10.4 In order to make a compute acceleration, did 2 string builder arrays to store a pre order sets.
3.2.10.5 in order to make a PCA POS acceleration, did 5 chars marching array to store a post order sets.

It seems the Nero INDEX [32], NLP and POS for the PCA separation with convolution kernel marching by using stepwise iterative differentiation got much higher sufficiency.

3.3 DETA Catalytic computing application

In order to demonstrate the importance of detailed logic, I began to integrate this logic concept into my YANGLIAOJING [11] 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 [20] 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[38], BLAINEZ), which is the only way to catalyze HUMANOID[20] DNA[14] evolutionary algorithm. And made a Detailed demonstration and summarized as follows

DETA Humanoid DNA[14] Details
3.3.1 Demonstration of differential algorithm of POS water valve with DETA fast segmentation[49];
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[20] DNA[14] evolutionary algorithm.

3.3.2 Demonstration of The 4th generation example demonstration of filtering sorting algorithm for LUO YAOGUANG s small peak calculation[50];
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[20] DNA[14] evolutionary algorithm can effectively smooth the peak of computation.

3.3.3 Demonstration of the second generation of differential TSP algorithm for LUO YAOGUANG Euler forest business travel ring[51].
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[20] DNA[14] evolutionary algorithm can effectively select the fastest algorithm module and cognitive module to do calculation in specific fields, and improve computing power.

3.3.4 Demonstration of the 7th generation example of Luo Yao s conditional differential sorting algorithm for light image strings[52];
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[9] logic, and form the purine element of INITON operation of DNA[14] peptide chain.
Demonstration influence: the guarantee of autonomous evolution mechanism of HUMANOID[20] DNA[14] evolutionary

algorithm.

3.3.5 Demonstration of PLSQL differential compiler for DETA Socket[3] stream programmable database engine[53];
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[20] DNA[14] 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[11]. Over the years, thinking, what is the final expression of Detailed logic?

4 DETA Finding INITONS

What is the final expression of Detailed logic? I have never stopped exploring, and I have always been absolutely focused. From 2018 to 2019, the final expression of logic refinement must not be as simple as AOPM[10] and VPCS[9]. VPCS[9] is just a refinement layer of AOPM[10], so how can VPCS[9]be refined?

4.1 DETA Finding INITONS history

What's under VPCS[9]? What is the essence of a function? At school, the primitives of DNA[14] are ACGTU purine and pyrimidine, the primitives of back-end architecture are VPCS[9], the primitives of thing logic are AOPM[10], 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?

4.2 DETA Finding INITONS 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 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[26]'s time flow form.

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[14]. Seems began to refine my sorting algorithm, word segmentation algorithm, ETL[2], YANGLIAOJING[11], etc., and found one thing in common. All my works were refined to the rational function level that could be 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.

4.3 DETA C Finding INITONS application

With this in mind, where determined a ternary mapping coding mode of DNA[14] to ETL[2] neuron nodes, AOPM[10] -VPCS- IDUC 3D coding mode. 4*4*4 Then each primitive is a 64-bit space, which is the computing primitive where looking for decades.

5 DETA DNA[14] DECODING[18]

If the AOPM[10] -VPCS- IDUC 3D computational neuron mapped by DNA[14] IDUC is established, how to decode it? about YANGLIAOJING[11]! It is the only way that at present to construct the system of YANGLIAOJING[11] and demonstrate this idea and technology. It is still the absolute focus of that sentence.

5.1 DETA DNA[14] DECODING[18] history

At present, what is that DNA[14] 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[20] data DNA[14] with IDUC unit can also write hundreds of thousands of business transaction processing logic of AOPM[10] VPCS. these two logics do not conflict.

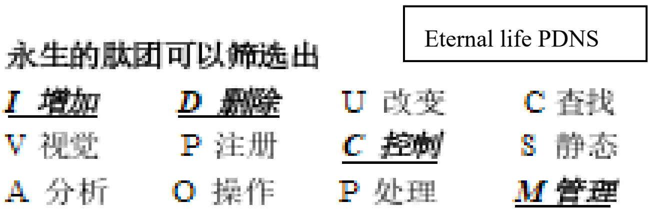
5.2 DETA DNA[14] DECODING[18] development

These two logics do not conflict. Are they one? 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[10] -VPCS- IDUC seems to explain all the answers I want.

5.3 DETA DNA[14] DECODING[18] application

In order to overthrow my argument, look for arguments everywhere to attack this argument. First, I found the topic of eternal life. According to AOPM[10] -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! Have been searching for answers to all the problems for decades from AOPM[10] -VPCS- IDUC, so started mapping and coding as follows:

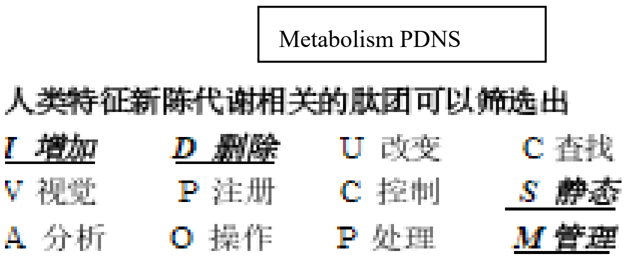
I	INCREMENT/ ADD	**D**	DECREMENT/FILTER	**U**	UPDATE	**C/Q**	QUERY/CHECK
V	VITIONARY/FEEL	**P/E**	EXECUTE	**C**	CONTROL	**S**	SET/STATIC
A	ANALYSIS	**O**	OPERATION	**P**	PROCESS	**M**	MANAGEMENT



Eternal life PDNS

永生的肽团可以筛选出

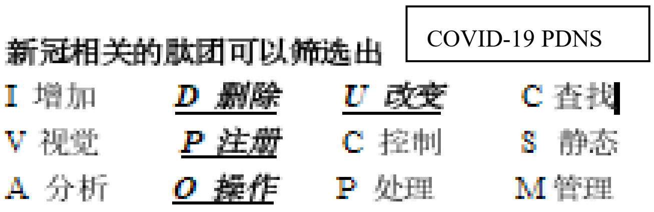
I 增加	***D 删除***	U 改变	C 查找
V 视觉	P 注册	***C 控制***	S 静态
A 分析	O 操作	P 处理	***M 管理***



Metabolism PDNS

人类特征新陈代谢相关的肽团可以筛选出

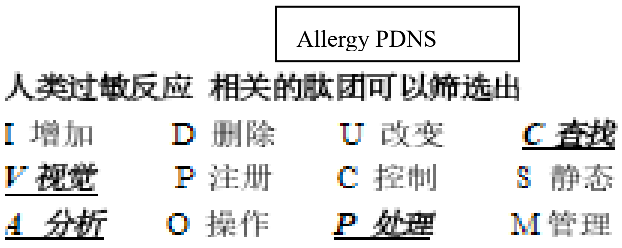
I 增加	***D 删除***	U 改变	C 查找
V 视觉	P 注册	C 控制	***S 静态***
A 分析	O 操作	P 处理	***M 管理***



COVID-19 PDNS

新冠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



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

DNA[14] theorem:
The essence of DNA[14] is a combination Indexing[32] link list of four meta-operations of adding, deleting modifying and Querying data.
10-04-2020 DC

6 IDUC DNA[14] and Its Applications

These are all the later stories. The application is too wide. First of all, my ETL[2] began to expand in the three-dimensional direction to better serve medicine. Secondly, virus immunology and immortal virus exploration will never stop. Why ETL[2] is used as the expansion point is inspired by my OSGI[24] paper on October 17, 2013. It is as follows: The Darwin[15]'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[15]'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[22][17] 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, 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.

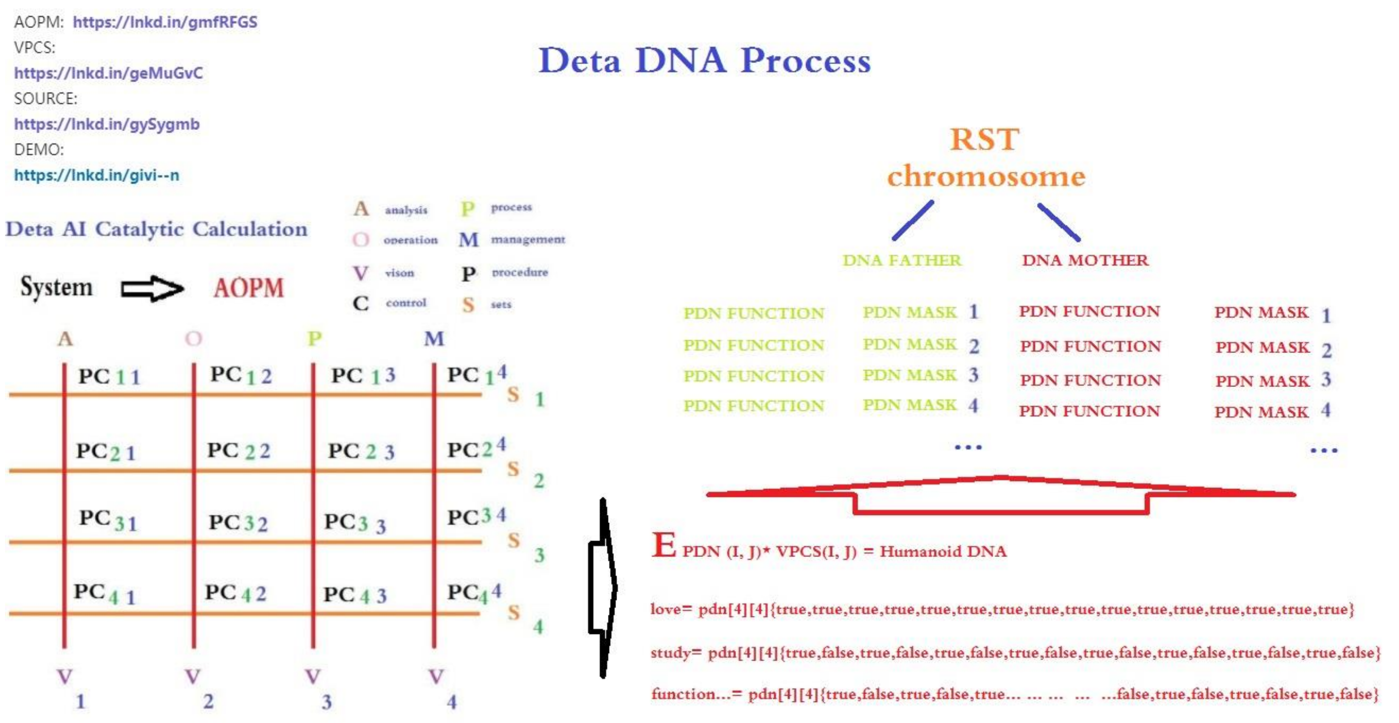
Through this description, it only proves that the greatest advantage of any language is only reflected in its creative theory and thought at the beginning of its birth. Therefore, JAVA and C# are not comparable at all. Because their original creative theory, system and ideological structure are different. 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[24] and Felix. The

OSGI[24] idea of KNIME[23] is consistent with that of LIFERAY. Although the API design style is different, the effect is very thick. Biology needs Darwin[15]'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[22][17] platform.

Now I have enough confidence to continue to focus on the argument of making ETL[2] mapped by my DNA[14] code with evolutionary system reuse the perfect guarantee requirement persistence. On how to use ETL[2] 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 See Figure 6-1



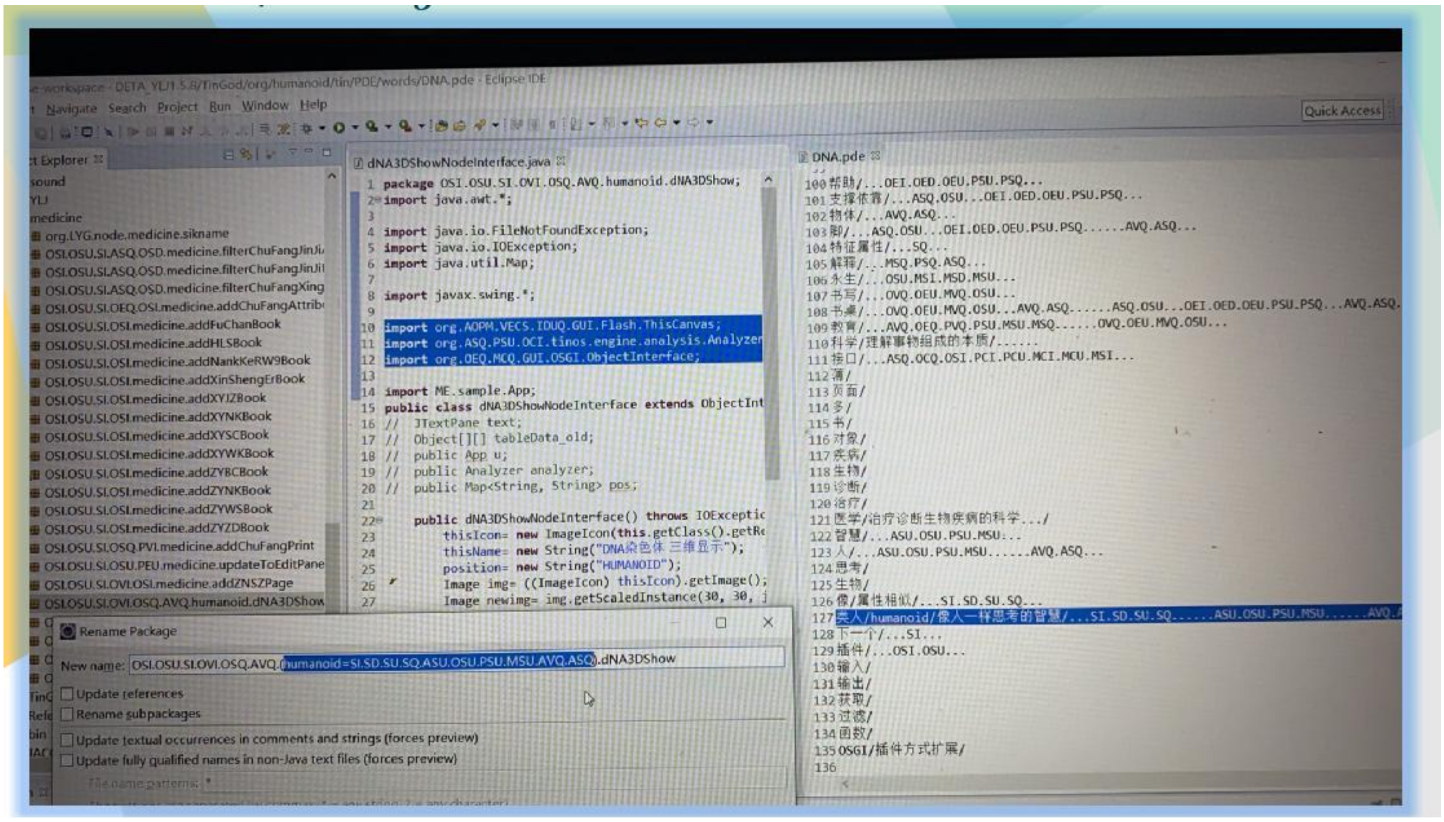
```
love= pdn[4][4]{true,true,true,true,true,true,true,true,true,true,true,true,true,true,true,true}
study= pdn[4][4]{true,false,true,false,true,false,true,false,true,false,true,false,true,false,true,false}
function...= pdn[4][4]{true,false,true,false,true... ... ...  ...  ...false,true,false,true,false,true,false}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

Figure 6-1

Exciting, ETL[2] node three-dimensional classification. this vocabulary

7 IDUC VPCS AOPM[10] 3D Nero Cell and Its Applications

These are the following words. ETL[2] 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[14] mapping of DETA IDUC. This is the real HUMANOID[20] independent thinking way that I can understand.
True Instance for <YANGLIAOJING>



e-workspace - DETA_YLJ1.5.8/TinGod/org/humanoid/tin/PDE/words/DNA.pde - Eclipse IDE

```
sound
YLJ
medicine
org.LYG.node.medicine.sikname
OSI.OSU.SI.ASQ.OSD.medicine.filterChuFangJinJi
OSI.OSU.SI.ASQ.OSD.medicine.filterChuFangJinJi
OSI.OSU.SI.ASQ.OSD.medicine.filterChuFangXing
OSI.OSU.SI.OEQ.OSI.medicine.addChuFangAttrib
OSI.OSU.SI.OSI.medicine.addFuChanBook
OSI.OSU.SI.OSI.medicine.addHLSBook
OSI.OSU.SI.OSI.medicine.addNankKeRW9Book
OSI.OSU.SI.OSI.medicine.addXinShengErBook
OSI.OSU.SI.OSI.medicine.addXYJZBook
OSI.OSU.SI.OSI.medicine.addXYNKBook
OSI.OSU.SI.OSI.medicine.addXYSCBook
OSI.OSU.SI.OSI.medicine.addXYWKBook
OSI.OSU.SI.OSI.medicine.addZYBCBook
OSI.OSU.SI.OSI.medicine.addZYNKBook
OSI.OSU.SI.OSI.medicine.addZYWSBook
OSI.OSU.SI.OSI.medicine.addZYZDBook
OSI.OSU.SI.OSQ.PVI.medicine.addChuFangPrint
OSI.OSU.SI.OSU.PEU.medicine.updateToEditPane
OSI.OSU.SI.OVI.OSI.medicine.addZNSZPage
OSI.OSU.SI.OVI.OSQ.AVQ.humanoid.dNA3DShow
```

dNA3DShowNodeInterface.java

```java
package OSI.OSU.SI.OVI.OSQ.AVQ.humanoid.dNA3DShow;
import java.awt.*;

import java.io.FileNotFoundException;
import java.io.IOException;
import java.util.Map;

import javax.swing.*;

import org.AOPM.VECS.IDUQ.GUI.Flash.ThisCanvas;
import org.ASQ.PSU.OCI.tinos.engine.analysis.Analyzer
import org.OEQ.MCQ.GUI.OSGI.ObjectInterface;

import ME.sample.App;
public class dNA3DShowNodeInterface extends ObjectInt
//  JTextPane text;
//  Object[][] tableData_old;
//  public App u;
//  public Analyzer analyzer;
//  public Map<String, String> pos;

    public dNA3DShowNodeInterface() throws IOExceptic
        thisIcon= new ImageIcon(this.getClass().getRe
        thisName= new String("DNA染色体 三维显示");
        position= new String("HUMANOID");
        Image img= ((ImageIcon) thisIcon).getImage();
        Image newimg= img.getScaledInstance(30, 30, j
```

DNA.pde

```
帮助/...OEI.OED.OEU.PSU.PSQ...
支撑依靠/...ASQ.OSU...OEI.OED.OEU.PSU.PSQ...
物体/...AVQ.ASQ...
脚/...ASQ.OSU...OEI.OED.OEU.PSU.PSQ......AVQ.ASQ...
特征属性/...SQ...
解释/...MSQ.PSQ.ASQ...
永生/...OSU.MSI.MSD.MSU...
书写/...OVQ.OEU.MVQ.OSU...
书桌/...OVQ.OEU.MVQ.OSU...AVQ.ASQ......ASQ.OSU...OEI.OED.OEU.PSU.PSQ...AVQ.ASQ.
教育/...AVQ.OEQ.PVQ.PSU.MSU.MSQ......OVQ.OEU.MVQ.OSU...
科学/理解事物组成的本质/......
接口/...ASQ.OCQ.OSI.PCI.PCU.MCI.MCU.MSI...
薄/
页面/
多/
书/
对象/
疾病/
生物/
诊断/
治疗/
医学/治疗诊断生物疾病的科学.../
智慧/...ASU.OSU.PSU.MSU:..
人/...ASU.OSU.PSU.MSU......AVQ.ASQ...
思考/
生物/
像/属性相似/...SI.SD.SU.SQ...
类人/humanoid/像人一样思考的智慧/...SI.SD.SU.SQ......ASU.OSU.PSU.MSU......AVQ.A
下一个/...SI...
插件/...OSI.OSU...
输入/
输出/
获取/
过滤/
函数/
OSGI/插件方式扩展/
```

Rename Package

New name: OSI.OSU.SI.OVI.OSQ.AVQ.(humanoid=SI.SD.SU.SQ.ASU.OSU.PSU.MSU.AVQ.ASQ).dNA3DShow

☐ Update references
☐ Rename subpackages
☐ Update textual occurrences in comments and strings (forces preview)
☐ Update fully qualified names in non-Java text files (forces preview)
File name patterns: *

Figure 6-2

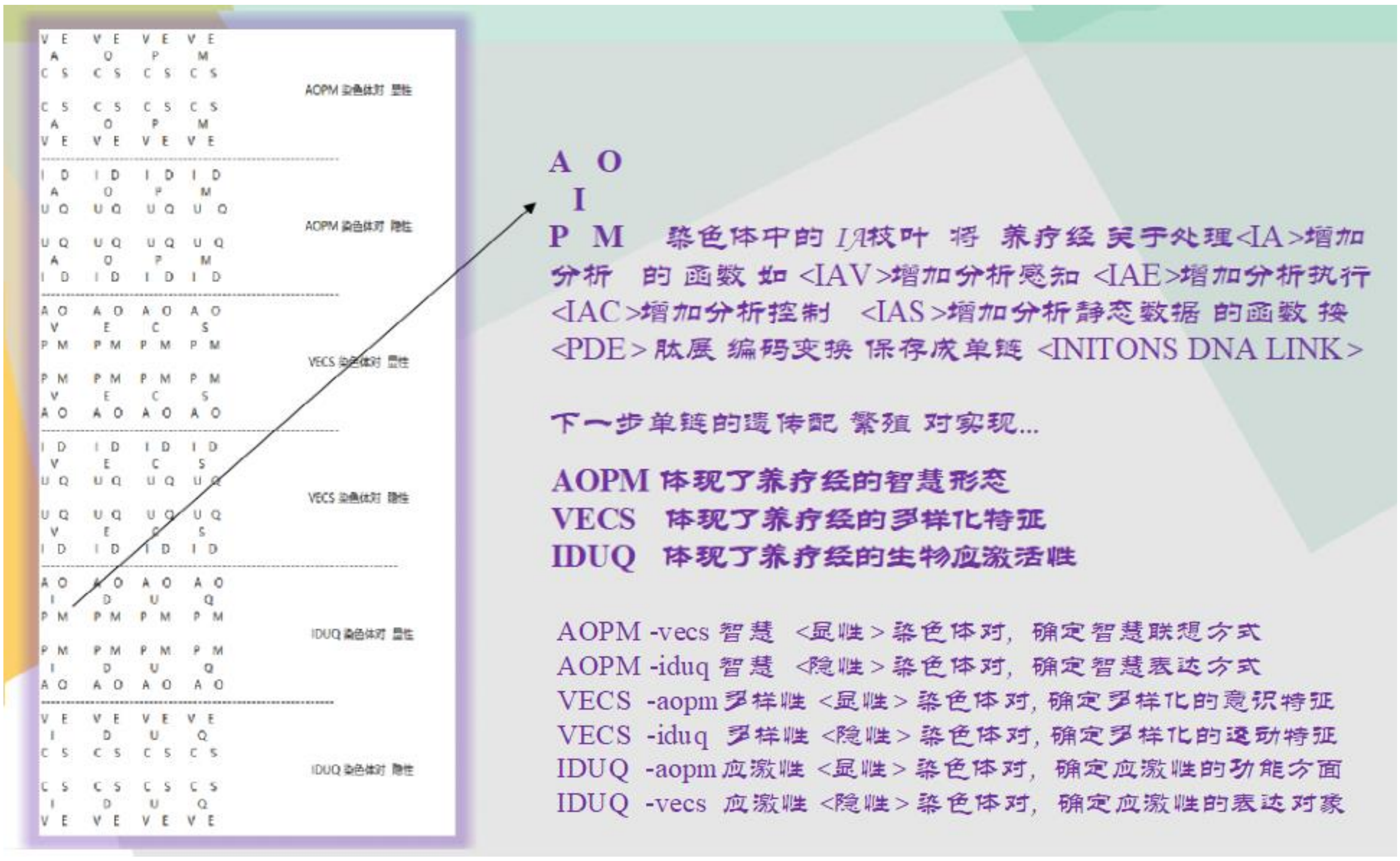
See Figure 6-2 The first INDEX[32] application idea of DNA[14] coding manual in human history.
will let Org.lyg.node.medcine.addchufangattributeH.jar where change to
Org.node.a.v.c.u.medcine.addchufangattributeH.jar

This A.V.C.U.. will form a DNA[14] mapping system code for analyzing visual control changes, which is convenient for future evolutionary optimization tests. After that, I will systematically encode these ETL[2] INDEX[32] mapping sets into DNA[14] INDEX[32] chains for YANGLIAOJING[11]. The ideas given to me in this paper are all creative ideas. Thank you for everything. I can first design AOPM[10] VPCS[9] 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[32] mode, even though it is not the final INDEX[32] of organic DNA[14] of human beings, has become the first mapping execution mode of HUMANOID[20] artificial design representing evolutionarily encoded DNA[14] as below Figure 6-3

.



染色体中的 IA枝叶 将 养疗经 关于处理<IA>增加分析 的 函数 如 <IAV>增加分析感知 <IAE>增加分析执行 <IAC>增加分析控制 <IAS>增加分析静态数据 的函数 按 <PDE>肽展 编码变换 保存成单链 <INITONS DNA LINK>

下一步单链的遗传配 繁殖 对实现...

AOPM 体现了养疗经的智慧形态
VECS 体现了养疗经的多样化特征
IDUQ 体现了养疗经的生物应激活性

AOPM -vecs 智慧 <显性>染色体对, 确定智慧联想方式
AOPM -iduq 智慧 <隐性>染色体对, 确定智慧表达方式
VECS -aopm 多样性 <显性>染色体对, 确定多样化的意识特征
VECS -iduq 多样性 <隐性>染色体对, 确定多样化的运动特征
IDUQ -aopm 应激性 <显性>染色体对, 确定应激性的功能方面
IDUQ -vecs 应激性 <隐性>染色体对, 确定应激性的表达对象

Figure 6-3 Chromosome INDEX[32] classify

Not The End

In this paper, Author 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[14] is a combination Indexing[32] link list of four meta-operations of adding, deleting, modifying and Querying data. By the way: Execution mode of neuron calculation: a neuron time series calculation chain of specifical function calculation by mapping of DNA[14] coding INDEX[32] reflection; The essence of adapting to the environment is that DNA[14] coding Indexes[32] map related neurons compiler link to achieve better addition, deletion, modification and query the environment data; The essence of HUMANOID[20] evolution offspring: the offspring produced by optimizing the hybridization of the same coding logic part in the DNA[14] encoding INDEX[32] chain mapped and retained by the neuron calculation method of data efficient processing

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.

8 Refer:

1. DETA PARSER, https://github.com/YAOGUANGluo/DETA_Parser
2. DETAETL, https://github.com/YAOGUANGluo/ETL_Unicorn
3. DETA Socket DB, https://github.com/YAOGUANGluo/DETA_DataBase
4. DETA TOP SORT, https://github.com/YAOGUANGluo/sort

5. SOFTWARE ENGINEERING https://baike.sogou.com/kexue/d10131.htm?ch=fromsearch
6. MVC, https://baike.sogou.com/v25227.htm?fromTitle=MVC
7. MVP, https://baike.sogou.com/v70887934.htm?fromTitle=MVP%E6%A8%A1%E5%BC%8F
8. ORACLE, https://baike.sogou.com/v110535.htm?fromTitle=oracle
9. VPCS, YAOGUANG. Luo
, https://github.com/YAOGUANGluo/DETA_Resource/blob/master/VPCS-Method_V1.1.doc
10.AOPM[10], _
https://github.com/YAOGUANGluo/DETA_Resource/blob/master/AOPM[10]%20System%20On%20VPCS.doc
11. YANGLIAOJING, http://tinos.qicp.vip/DETA_HUARUIJI[12].html
12. HUARUIJI[12], http://tinos.qicp.vip/DETA_HUARUIJI[12].html
13. SONAR[13], _https://www.SONAR[12]lint.org/
14.DNA[14],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[14]&fromid=98123&fr=aladdin
15.Darwin[15],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
16.NERO CELL[16]
,https://baike.baidu.com/item/%E7%A5%9E%E7%BB%8F%E5%85%83/674777?fr=aladdin
17.AI[17], https://baike.baidu.com/item/%E4%BA%BA%E5%B7%A5%E6%99%BA%E8%83%BD/9180
18.EN-DECODING[18], https://baike.baidu.com/item/%E7%BC%96%E7%A0%81%E8%A7%A3%E7%A0%81
19.PARAL[19]
COMPUTING, https://baike.baidu.com/item/%E5%B9%B6%E8%A1%8C%E8%AE%A1%E7%AE%97
20.HUMANOID[20], https://baike.baidu.com/item/%E4%BB%BF%E7%94%9F%E4%BA%BA/5142593?fr=aladdin
21.Data Mining[21],https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%8C%96%E6%8E%98/216477
22.UNICORN AI[22][17], https://github.com/YAOGUANGluo/ETL_Unicorn
23.KNIME[23]
,https://www.baidu.com/link?url=g_8i8yfDrvNMqYfN6A9z_XoIc49s8yqzHgpYn-JCBIi&wd=&eqid=88da123b0000be28000000065f7eb5df
24.OSGI[24], https://www.oschina.net/p/OSGI[24]?hmsr=aladdin1e1
25.DML[25],
https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E6%93%8D%E7%BA%B5%E8%AF%AD%E8%A8%80/10826467?fromtitle=DML[25]&fromid=10035808&fr=aladdin
26.John Von Neumann[26],
https://baike.baidu.com/item/%E7%BA%A6%E7%BF%B0%C2%B7%E5%86%AF%C2%B7%E8%AF%BA%E4%BE%9D
27.FFT[27] KERNEL https://baike.baidu.com/item/%E8%9D%B6%E5%BD%A2%E8%BF%90%E7%AE%97/4756906
28.THREAD[28], https://baike.baidu.com/item/%E5%A4%9A%E7%BA%BF%E7%A8%8B
29.BLUE TOOTH[29], https://baike.baidu.com/item/%E8%93%9D%E7%89%99
30.PDN LINK[30], https://baike.baidu.com/item/%E8%82%BD%E9%93%BE/8625112?fr=aladdin
31.SDLC[31]: “”, https://en.wikipedia.org/wiki/Systems_development_life_cycle
32.INDEX[32], https://baike.sogou.com/v65431335.htm?fromTitle=%E7%B4%A2%E5%BC%95
33.SLANG[33], https://baike.sogou.com/v18863.htm?fromTitle=%E6%88%90%E8%AF%AD
34.SCRIPT LANGUAGE[34],

https://baike.sogou.com/v230334.htm?fromTitle=%E8%84%9A%E6%9C%AC%E8%AF%AD%E8%A8%80
35.PROSCRIPTION[35], https://baike.sogou.com/v5065331.htm?fromTitle=%E5%A4%84%E6%96%B9
36.XUESEN QIAN[36],
https://baike.sogou.com/v237588.htm?fromTitle=%E9%92%B1%E5%AD%A6%E6%A3%AE
37.DIFFERENTAL[37], https://baike.sogou.com/v1966327.htm?fromTitle=%E5%BE%AE%E5%88%86
38.NEWTON[38], https://baike.sogou.com/v88517.htm?fromTitle=%E7%89%9B%E9%A1%BF
49.Leibniz[39], https://baike.sogou.com/v10692238.htm?fromTitle=%E5%B8%83%E8%8E%B1%E5%B0%BC%E5%85%B9
40.RECURATION[40], https://baike.sogou.com/v3195520.htm?fromTitle=%E8%BF%AD%E4%BB%A3
41.POLYNOMIAL[41], https://baike.sogou.com/v445974.htm?fromTitle=%E5%A4%9A%E9%A1%B9%E5%BC%8F
42.OSS Book[42]: Eric Steven Raymond, 《The Cathedral and the Bazaar》
43.DETA parser source from GITEE: https://gitee.com/DETAChina/DETAParser[43], DETA parser source from GITHUB: https://github.com/YAOGUANGluo/DETA_Parser
44.DETA parser split method[44] record: https://github.com/YAOGUANGluo/DETA_Parser/issues/21,
DETA parser official demo: http://tinos.qicp.vip/data.html
45.DETA parser integrated products demo from GITHUB:
https://github.com/YAOGUANGluo/DETA_Medicine[45],
DETA parser integrated products demo from GITEE: https://gitee.com/DETAChina/DETA_Medicine[45],
DETA parser integrated products demo download link:
http://tinos.qicp.vip/download/HUARUIJI[12]Tm1.0.3.zip
46.Reflection on YAOGUANG's Peak Array Split Defect[46]1.0
https://github.com/YAOGUANGluo/DETA_Resource/blob/master/Reflection%20on%20YAOGUANG's%20Peak%20Array%20Split%20Defect1.0.pdf,
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
47.UNIX, https://baike.sogou.com/v5436069.htm?fromTitle=UNIX[47]
48.LINUX, https://baike.sogou.com/v807585.htm?fromTitle=LINUX[48]
49.Demonstration of differential algorithm of POS water valve with DETA fast segmentation;
https://gitee.com/DETAChina/DETAParser[43]/blob/master/wordSegment/org/tinos/engine/pos/imp/POSControllerImp.java
50.Demonstration of The 4th generation example demonstration of filtering sorting algorithm for LUO YAOGUANG s small peak calculation;
https://gitee.com/DETAChina/DataSwap/blob/dceeb0b06f726d640553964058d85b736354ac89/src/org/DETA/tinos/array/LYG4DWithDoubleQuickSort4D.java
51.Demonstration of the second generation of differential TSP algorithm for LUO YAOGUANG Euler forest business travel ring
https://gitee.com/DETAChina/Data_Prediction/blob/master/src/org/tinos/DETA/tsp/YAOGUANGLuoEulerRingTSP2D.java
52.Demonstration of the 7th generation example of Luo Yao s conditional differential sorting algorithm for light image strings;
https://gitee.com/DETAChina/DataSwap/blob/master/src/org/DETA/tinos/string/LYG4DWithChineseMixStringSort7D.java
53.Demonstration of PLSQL differential compiler for DETA Socket[3] stream programmable database engine;
https://gitee.com/DETAChina/DETA_PLSQL_DB/blob/master/java/org/lyg/db/plsql/imp/ExecPLSQLImp.java

9 Others:

GIT Thanks 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:
GIT Thanks 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.
GIT Thanks https://github.com/YAOGUANGluo/DETA_CACHE, this project is a rest calls cache by using VPCS TCP method. It seems very tiny and fast.
GIT Thanks https://github.com/YAOGUANGluo/DETA_BackEnd, this projects is a backend rest service by using VPCS.
GIT Thanks https://github.com/YAOGUANGluo/ETL_Unicorn,this, project is a Data Mining[21] ETL software by using VPC, similar with the KNIME[23]. Author makes high recommendation about this portrait.
Thanks GIT Hub, GITEE, Eclipse.org, Microsoft, Apache and Java. Google, Angular, JQUERY, Bootstrap, JSON, Oracle[8].
Thanks from the bottom of my deep heart.
Thanks to my family, my colleagues, and my customers. Common!