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.

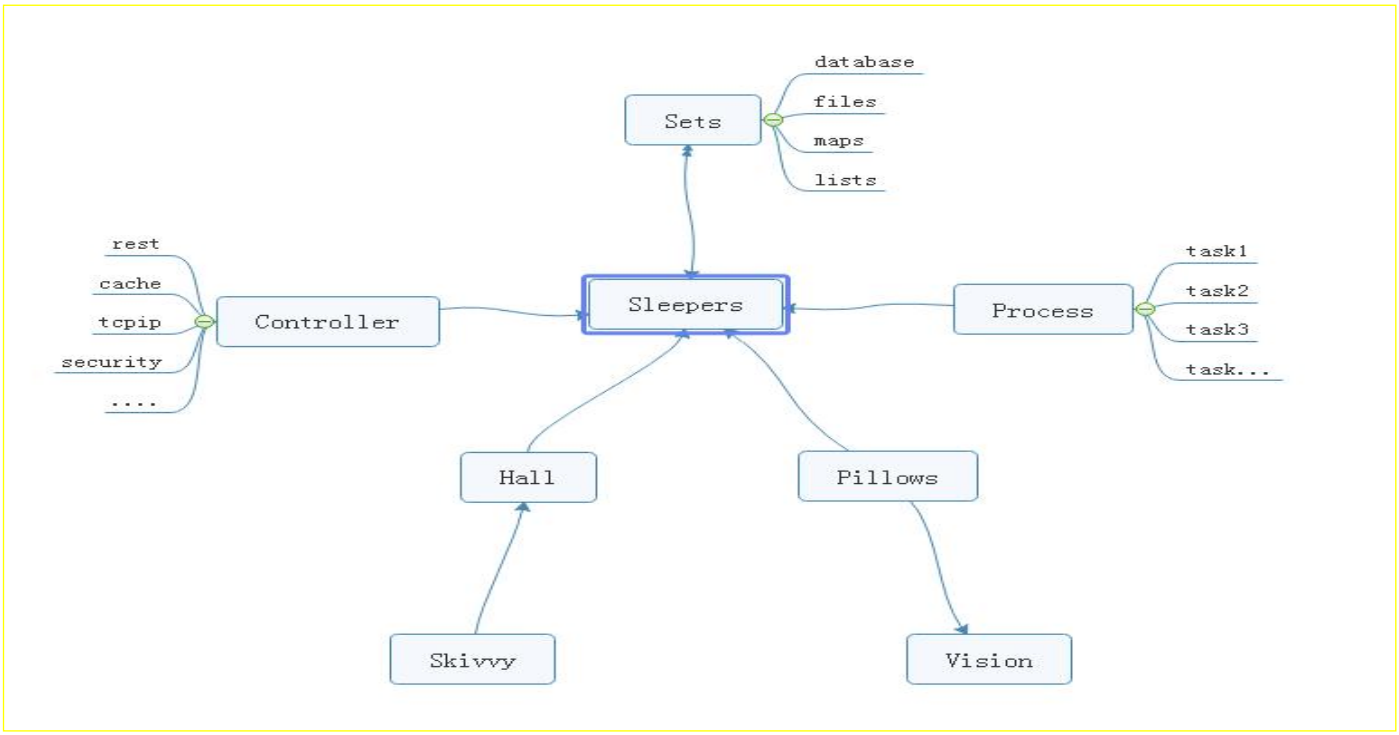

Figure 1-1 VPCS STAR MODEL

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.

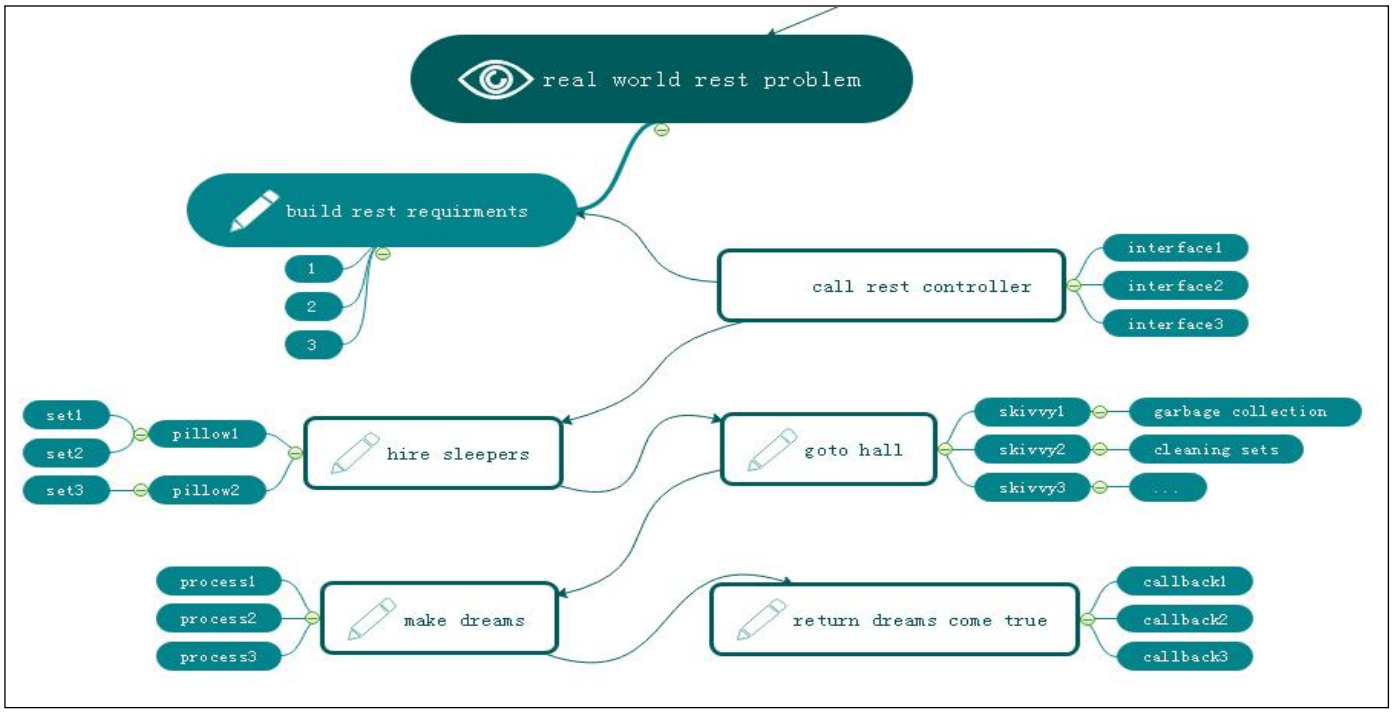

Figure 1-2 VPCS BACKEND MODEL

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.

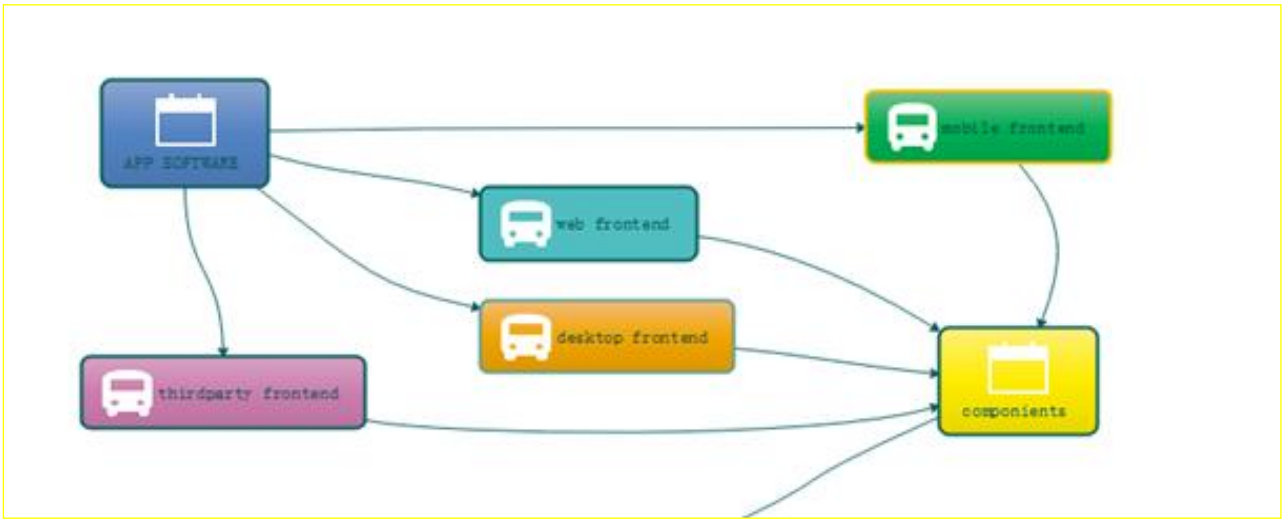

Figure 1-3 VPCS WORK WITH FRONTEND

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.

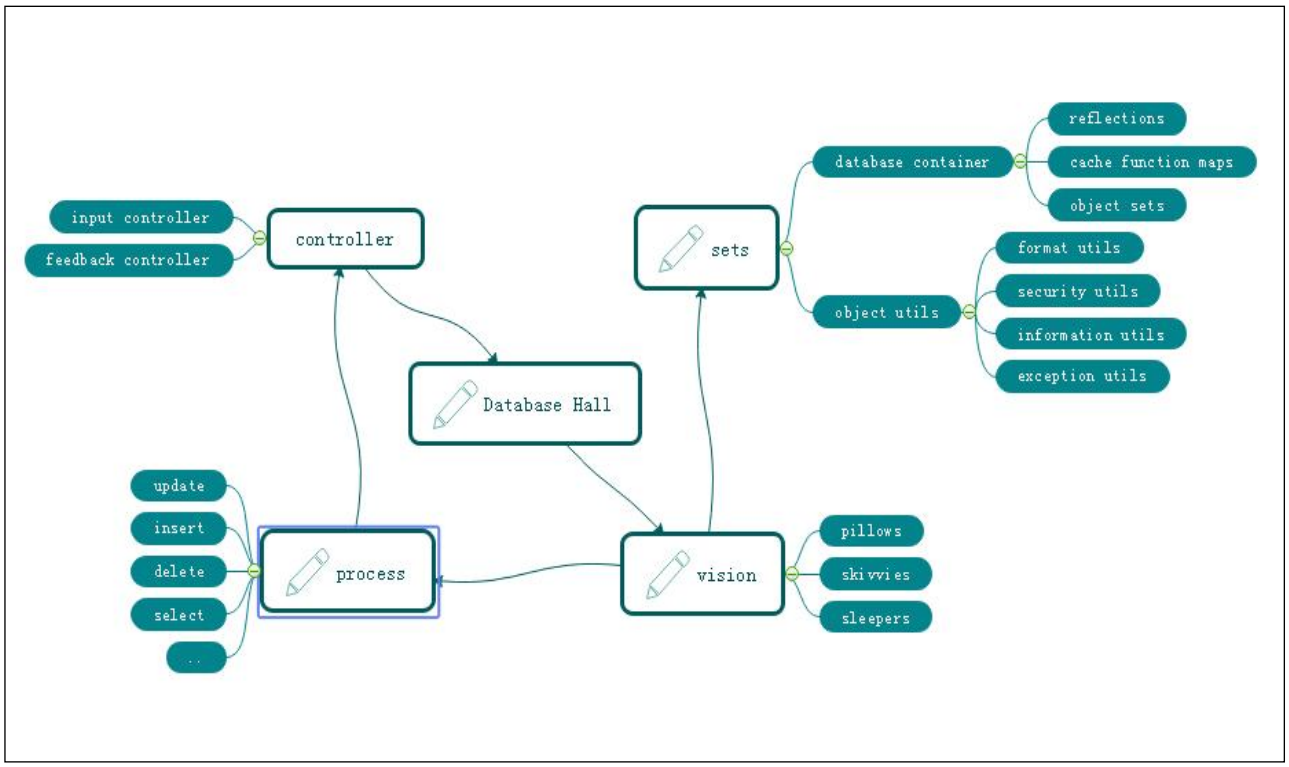

Figure 1-4 VPCS FOR DATABASE SYSTEM

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 singer 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.

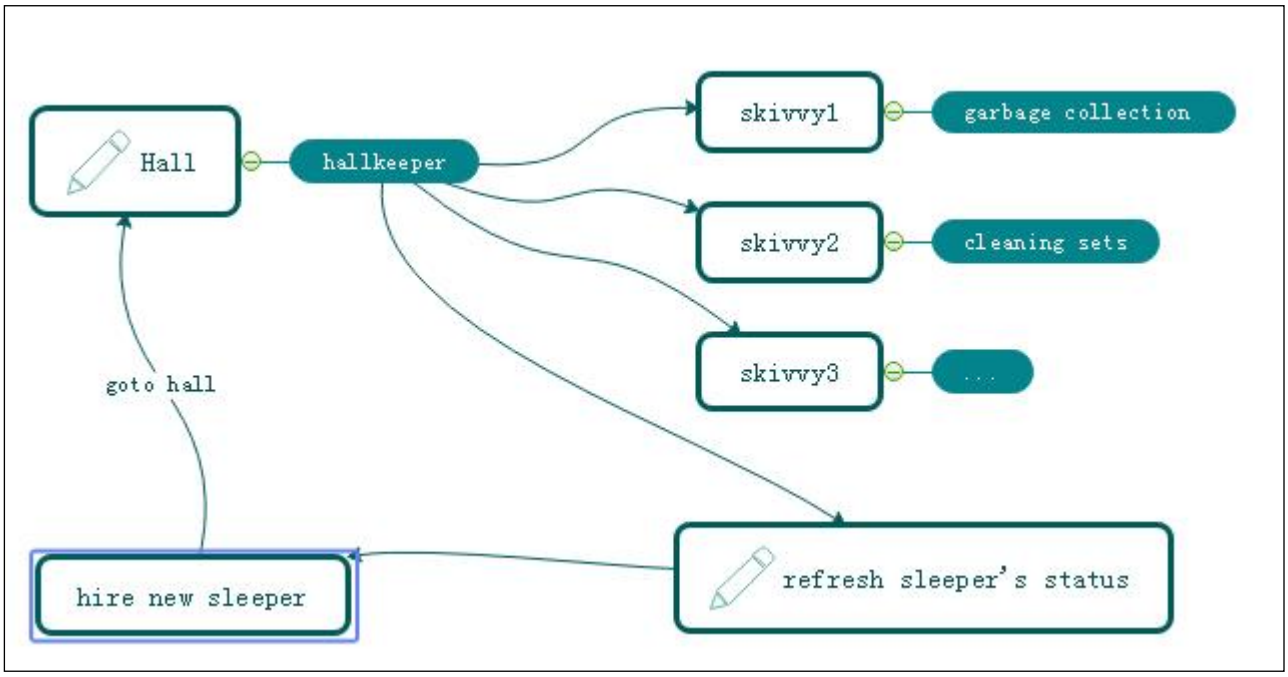

Figure 1-5 VPCS KERNEL

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.

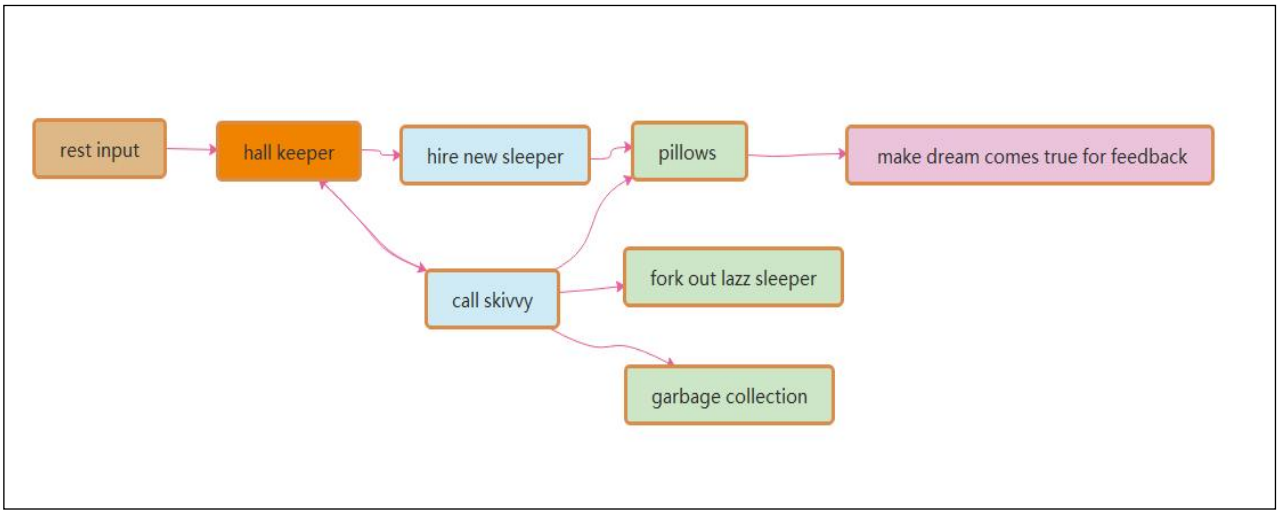

Figure 1-6 VPCS Sequence Diagram

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.

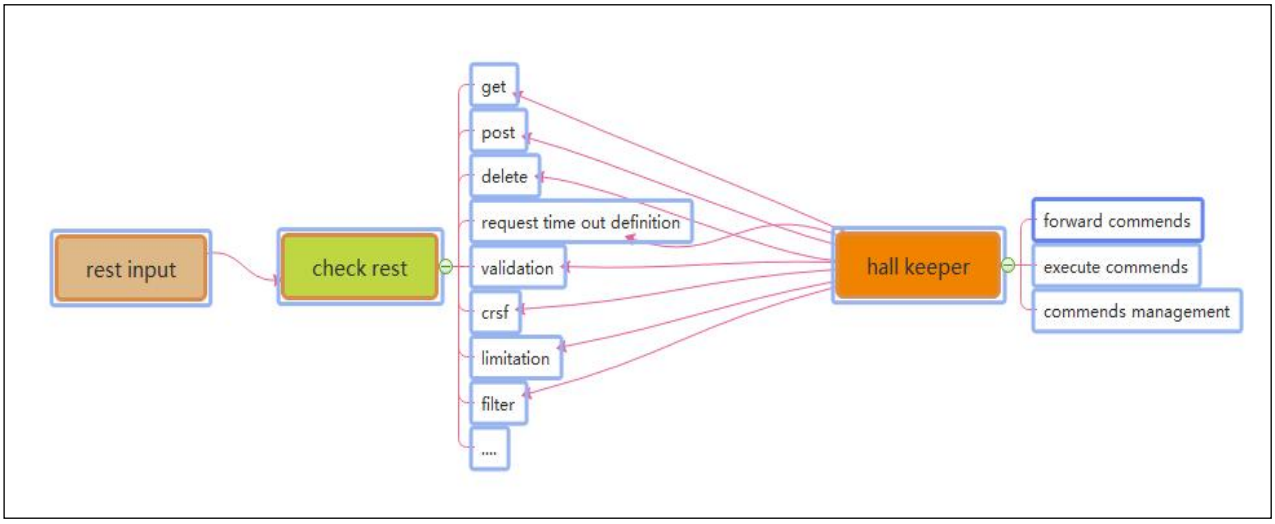

Figure 1-6-1 The Interaction Between Rest Call and Hall Keeper

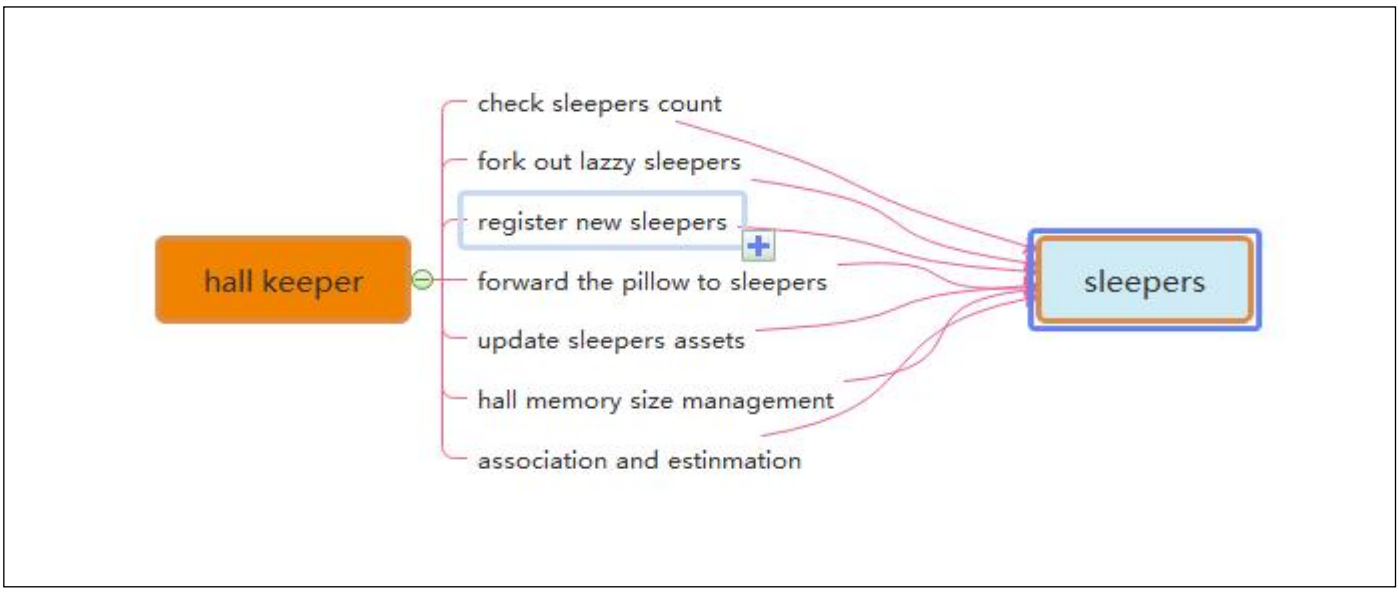

Figure 1-6-2 The Interaction Between Hall Keeper and Sleepers

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

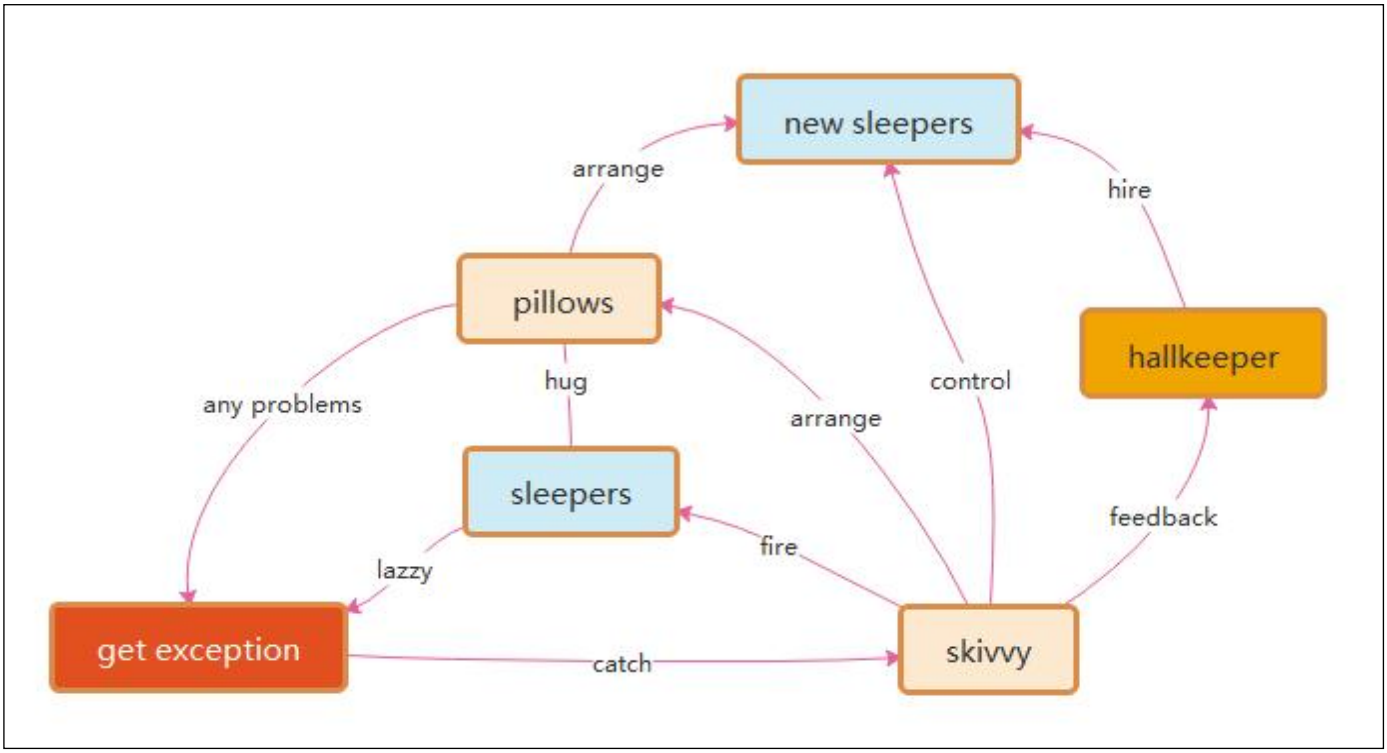

Figure 1-6-2-1 VPCS kernel

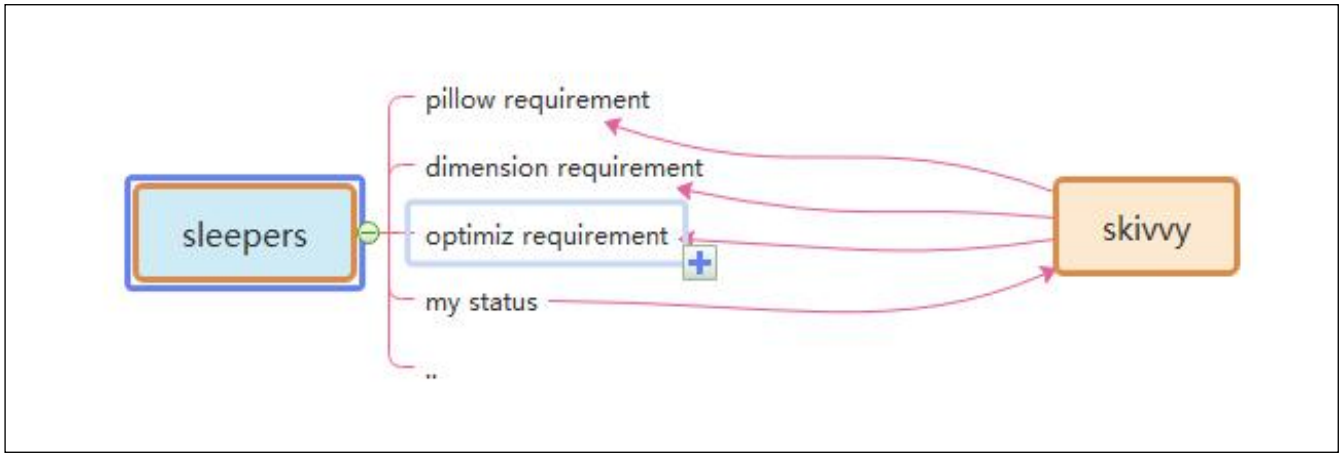

Figure 1-6-3 The Interaction Between Sleepers and skivvy

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.*

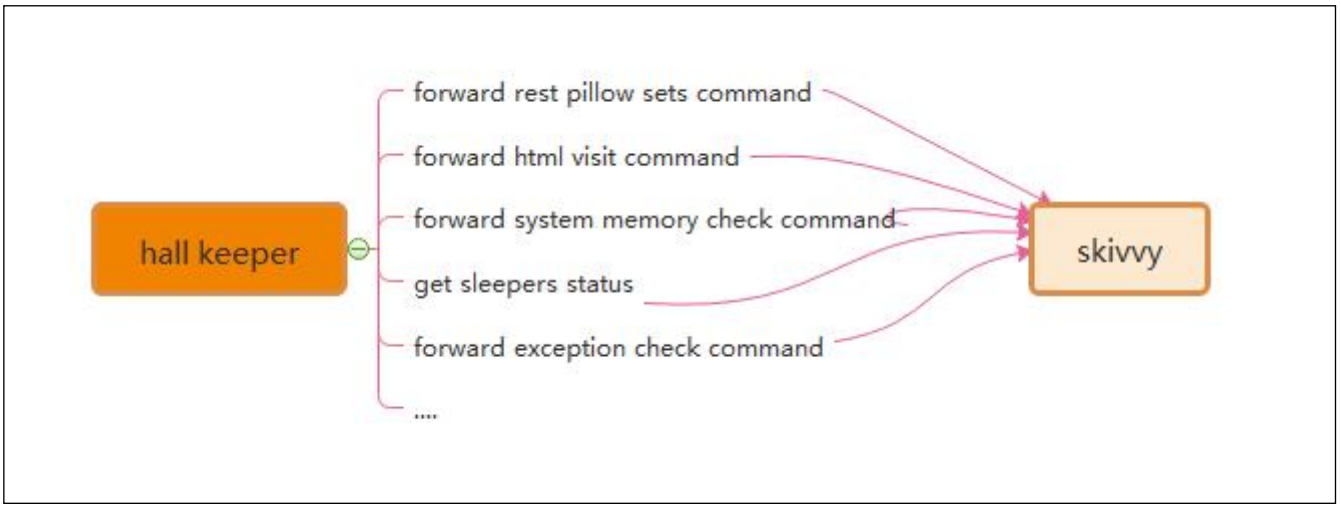

Figure 1-6-4 The Interaction Between Skivvy and Hall Keeper

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.

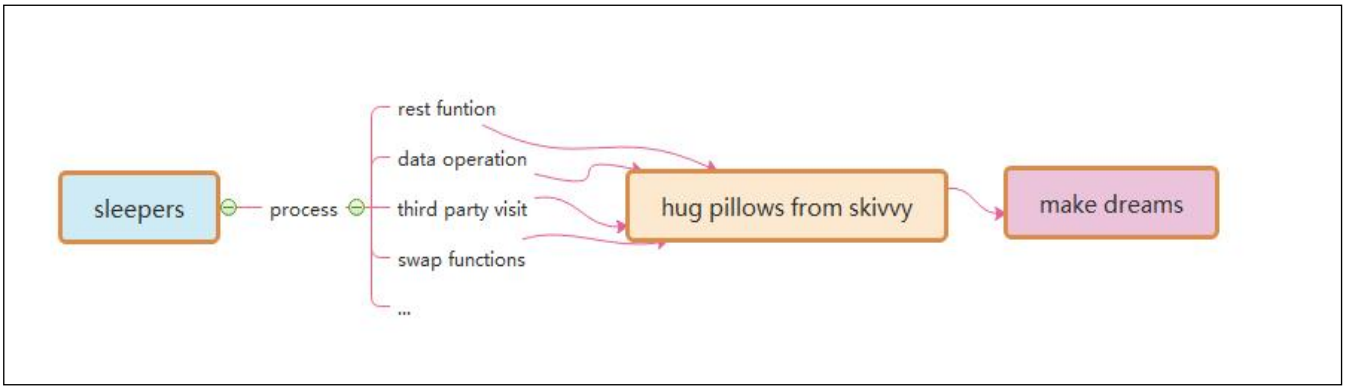

Figure 1-6-5 The Interaction Between Sleepers and Pillows

References

All guys will find the true source of the WEB link:

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.

Acknowledgement

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!