Assignment 3 : identify architectural tactics
3120102146 葛现隆

Tactics
(1) “Implement inter-service monitoring and alerting.”
Monitoring is a kind of fault detection which will enhance the availability. All that is needed to do is watching the running state of other system components.”If the service is overloading a dependent service, the depending service needs to know and, if it can’t back-off automatically, alerts need to be sent.” Yet, if operations can’t resolve the problem properly, it’s time to contact engineers.

(2) “Partition the service. ”
Partition is important for systems modifiability. Partition can reduce the size of a model, which will reduce the cost of modification. And it can also reduce coupling, “not bounded by any real world entity”.

(3) “One pod or cluster should not affect another pod or cluster”
This means we need to reduce coupling, by using Encapsulate, intermediary, Restrict Dependencies, Refactor, Abstract Common Services and so on. By all of these, we can stop cluster affecting each other, make them more independent, reduce the cost of modification and make the connection more efficient.

(4) “Keep things simple and robust.”
Simple is beautiful. High Testability need limit complexity, limit structural complexity.”Complicated algorithms and component interactions multiply the difficulty of debugging, deploying, etc. ” Simple sometimes is near stupid but is almost always better in a high-scale service. We need to avoid cyclic dependencies between components, isolate and encapsulate dependencies on the external environment and so on. Just make the system more simple and robust.

(5) “Analyse throughput and latency. ”
This is very important to systems performance(Tactics for Performance).Response is an important part of performance, and is measured by latency, deadlines in processing, throughput, jitter and so on.”Analysis of the throughput and latency of core service user interactions should be performed to understand impact. ” as well as other operations.

(6) “Understand access patterns. ”
We have learned many access patterns, like facade pattern, bridge pattern, mediator pattern,proxy pattern, factory pattern and so on. In those patterns, we apply abstraction or proper interface, that need us to consider “What impacts will this feature have on the rest of the infrastructure?” With proper access patterns, our system will be more efficient.