The Study on Capacity Enhancement of Distributed Systems Cloud Services

Xueping Liu[1,2]
1 Engineering Training Center
Shenyang Aerospace University
Shenyang, China
2 College of Automation Engineering
Nanjing University of Aeronautics and Astronautics
Nanjing, China
liuxueping024@163.com

Ji Wang
Dalian information center of the fiscal and taxation
Dalian, China
czj_wangji@dl.gov.cn

Abstract—With the Number of users in distributed systems clouds rapidly growing every day, increased workload, data and network traffic are seen in the clouds. The SNP (social-net processor) part would become the bottleneck probably as it interacts with social-net providers super frequently. This need to be evaluated pro-actively and coped with before the whole cloud gets crashed due to the huge increased burden. Adding new cloud with additional hardware and equipments is a way but quite costly. As a more practical approach, optimizing existing system to lift the capacity will be extremely beneficial. This involves improving parts of system infrastructure and workflow, fully utilizing cloud resources and balancing the cooperation among components such as SNP, Core Service, DB, Memcache etc. problem identification, performance or capacity modeling, solution design and implementation, and change impact assessment are the main work-streams.This paper will detail the useful principles and proven practices applied to the optimization.

Keywords—fuzzy face; low-quality images; shape features; ASM method

I. INTRODUCTION

Cloud Services is a group of mobile internet services provisioned to market[1,2]. With the number of user rapidly increasing, we used to continuously setup new clouds by purchasing hardware and equipment to accommodate them[3]. More than 100 production cloud physical server is currently a few K.This is too costly on application servers, data storage equipments, network, data center space, power and operation engineers[4,5]. In order to reduce cost, we initiated the Capacity Enhancement Project to increase our single child cloud's capacity[6].Currently, Cloud Computing is prevailing, it does provide IaaS(Infrastructure as a Service) solutions to reduce cost and enhance capacity by increasing the efficiency of hardware resources[7], but it require relatively more bigger effort, so it is a long run candidate solution. In current phase, we only focus on increasing the capacity by improving the application level[8], mostly from current application architecture, design, implementation aspects[9].

For any system, system optimization implements Use Cases by its internal components collaborating[10]with each other.

The division of components is System Structure.

The collaboration of components to implement an Use Case is System Behavior.

System's capacity can be quantified as the maximum load that can still keep KPIs(Key Performance Indicators) normal.

Software system structure domain model and System optimization model are as follows:

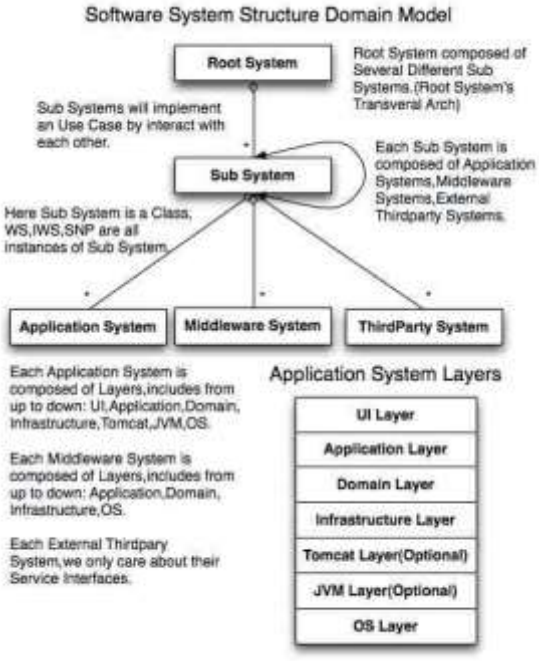

Figure 1. Software System Structure Domain Model

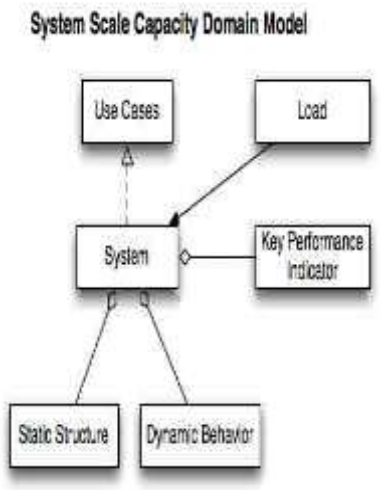

Figure 2. System Optimization Domain Model

II. RELATED WORK

To do system optimization, firstly, we need to profile current system status, including request counts, response time and resource consumption. After that, we can setup an optimization target based on current system status, the capacity and performance we can accept. Then, try to find the system bottleneck by stress test. Finally, find out the related solutions to eliminate the bottleneck, reduce system resource consumption and improve system performance.

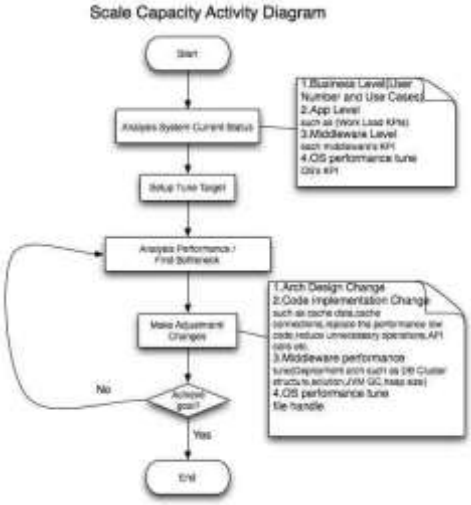

Figure 3. System Optimization Activity Diagram

Before system optimization, we should understand Cloud Services' main feature:

1) the integration of popular social networks to android device,

2) enable device users to connect friends more tightly and socially by keeping the users online of internet,

3) and get SN updates of his or her friends in time,

4) post SN updates such as statuses, photos real time and interact with friends.

For Cloud Services, to support each user, system will compute SN statuses, friendfeed, friends, snmail, comments changes by scheduler syncing,this work bring the most part load to system.Take Renk users for example, each user will has 24.2 works per hour for SNPSVC to do.We also need to profile the each SN types' change rate.

Provider: Type: Work

Name	Count	AVG	MIN	MAX
SNPWorker CommentsRequest	120	1837	0	16501
SNPWorker MailAction	2	134	0	184
SNPWorker RequestReauth	1	69	0	69
SNPWorker SetStatus	4	982	0	3720
SNPWorker StatusReaction	4	2853	0	10889
SNPWorker SyncComments	1929	10911	0	52941
SNPWorker SyncCommentsClient	1823	925	0	30092
SNPWorker SyncFriendFeed	8140	17514	0	84008
SNPWorker SyncFriendFeedClient	748	716	0	23124
SNPWorker SyncFriends	1373	9878	0	56353
SNPWorker SyncFriendsClient	94	932	0	17641
SNPWorker SyncMail	4531	8858	0	78325
SNPWorker SyncMailClient	6	83	0	158
SNPWorker SyncStatus	9548	14417	0	78830
SNPWorker SyncStatusClient	1926	629	0	68632

Figure 4. Renk Users

We realize that Quantify as KPIs under a Load.How to quantify KPIs, According to System Structure:

System performance can be quantified as a serial of KPIs as Sub Systems' KPIs.

Each sub system performance can be quantified as each Tier's KPIs.

Each tier performance can be quantified as each layer's performance.

As we are focus on doing optimization in the sub system SNPSVC, so we quantify its KPIs as

Application Tier
Application Layer
Works Submitted/second
Works Completed/second
Works Failed/second
Work Queue Size
Work execution time
OS Layer
CPU User Usage
CPU Load Average
Middleware Tier
MySQL DB
Inserts/second
deletes/second
updates/second
selects/second

KPIs' normal value:

Works Submitted, Works Completed and Works Failed: Work Failed number should be no more than 1% of Work Submitted number.

Work Queue Size: The Work Queue Size value is discrete, the normal state is work queue size fluctuate in a empiric value and it is no more than 10 works in usually case.

Work Execution Time: The normal state is work execution time fluctuate in a empiric value and it is no more than 1 sec in usually case.

CPU User Usage: CPU User Usage should be no more than 65% -- 70%.

CPU Load Average: CPU Load Average should have no more than 1--3 threads queued per processor.

Find Bottleneck by Stress Test:

In order to find bottleneck, stress test tool is need to emulate the real load post to system. For Cloud Services, as it need to use the SN provider's services, also need to implement fake SN provider services. In experience, we found DB is the system bottleneck.

Analyze and Find Solutions:

The optimization goal is increasing system capacity, so we will consider the solutions from below aspects.

1) Eliminate the system bottleneck by reducing DB load.

2) Reduce system resource consumption.

3) Improve system performance.

III. EXPERIMENTAL RESULTS

A. *Eliminate System DB Bottleneck*

1) *Apply Push Mechanism*

To eliminate system DB bottleneck, we checked the system each tier design, tried to find out if the unnecessary work stream nodes is exist which can be removed. Finally, we focus on the interaction between SNP and IWS.

In current system, the work stream of interaction between SNPSVC and IWS as below.

SNPSVC request changes from Social Network.

SNPSVC store changes into SNP DB.

SNPSVC notify IWS come to get the changes.

IWS request SNPSVC to get changes by querying SNP DB.

IWS store changes in transfer queue.

IWS response to SNPSVC.

SNPSVC receive the response, update sync-anchor and last-updated-timestamp, and delete the changes in SNP DB.

We consider that SNPSVC can push the changes to IWS directly.

After apply "push" mechanism, the work stream of interaction between SNPSVC and IWS as below.

SNPSNV request changes from Social Network.

SNP push changes to IWS directly.

IWS store changes in transfer queue.

IWS response to SNPSVC.

SNPSVC receive the response, update sync-anchor and last-updated-timestamp.

In a word, the "push" mechanism remove the unnecessary SNP DB operations, including insert, query and delete. It reduces the DB load and improve system performance significantly.

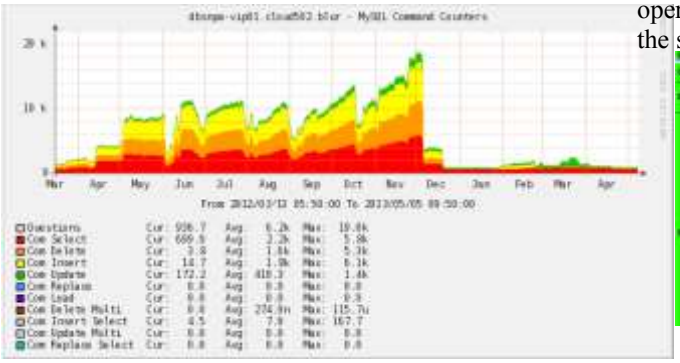

Figure 5. Sub System Interactions Optimization Effect

2) *Use Memcached instead of DB*

a)Reduce one work's execution time, improve the system performance.

b)Reduce db select operations, make db is not the bottleneck any more.

3) *Remove Unnecessary DB Synchronous Code Block*

This reduces the waiting time for DB synchronous resource, improves DB performance.

B. *Reduce System Resource Consumption*

1) *Prolong Polling Intervals*

To reduce system resource consumption, the most effective way is reducing the number of works in unit time. So by trade off between cost with Qos, product team approve the downgrade status, snmail, friendfeed's Qos from device user receive status updates no later than 5 mins to 20 mins, receive friendfeed updates no later than 10 mins to 30 mins, snmail's from 15 mins to 30 mins.

This will reduce works per user per hour from 24.2 to 9.2, make system capacity enhance 2.6 times.

2) *Apply Suspend Mechanism for Inactive Users*

For the users who did not login device beyond 48 hours, system will set this user to "suspend" state. It means that scheduler will not sync for this user until the users is resumed.

This reduce the number of work, so reduce system resource consumption.

3) *Limit the Minimum Intervals between Two Sync-Now Works of User*

Sync now requests are triggered by users from device. If two continuous such requests come to servers within short intervals, the second request will not get updates most possibly. To avoid wasting resource, the unnecessary second request will be discarded. Increasing the effective interval values can be helpful.

C. *Improve System Performance*

1) *Cache SN Provider API Call Result into Memcached to Reduce SN Provider API Calls*

This reduce one work's execution time, increase the system's throughput, works completed number per sec.

2) *Downgrade Log Level to Reduce File IO Operations*

Our system only prints DEBUG level or more higher level log in log file. So for some unnecessary log, we downgrade them to TRACE level. This can reduce file IO operations and reduce one work's execution time, increase the system's throughput, works completed number per sec.

Figure 6. the KPIs' comparison

From the KPIs' comparison between "Optimized" and "Not Optimized", it can be found that we have eliminated system DB bottleneck, reduced system resource consumption and improved system performance, so the optimization effect is significant.

IV. CONCLUSION

System optimization is a very complicated work, it require analyzing all levels. To make sure high performance, it require to take actions in nearly every phases, including requirement phase, design phase, implement phase, operation phase, and require efforts of architect, designer, developer and QA and require the scientific methods.

A. In Requirement Phase

For architect, need to

Define KPIs for performance analysis and monitoring.

Define requirements that log performance analysis and load model needed data when implement features.

Define requirements that expose KPI as JMX for performance analysis when implement features.

B. In Design Phase

For architect, need to carefully design the interactions and interfaces, reduce unnecessary interactions and works.

For designer, when do detail design, DB is always a bottleneck, we can reduce this bottleneck by using Memcache to reduce DB operations.

C. In Implementation Phase

For developer, need to

Meet the performance related requirements defined by Architect.

Make sure the implementation code is high efficiency.

Comply with the rule that use cache instead of

DB if needed

D. In Stress Test Phase

Tester will need to first profile the load model, make sure the model is close to real case and scenario and make a stress test tool to do stress test.

REFERENCES

[1] JIANG Wei-yin,LI Bin,LING Li, "Research on Data Consistency and Concurrency Optimization of Distributed System," China. Computer Engineering,2012(2),vol.38,No.4.

[2] Sotomayar B,Montero R S, Lorente I M,et al.,"Virual Infrastructure Management in Private and Hybrid Clouds[J] ," IEEE internet Computing,2009,13(5):14-22.

[3] Jaechun N. Data Consistency Protocol for Distributed File Systems[C],Proc. of Conf. on Intelligent Data Acquisition and Advanced Computing Systems:Technology and Applications.Rende, Italy:[s.n.],2009.

[4] Choi Sung-Chune,Choi Min-Seuk,Lee Chun-Kyeong,et al.Distributed Lock Manager for Distributed File System in Shared-disk Environment[C],Proc.of the 10th Int'l Conference on Computer and Information Technology.Bradford, UK:[s.n.],2010.

[5] Baliga J,Ayre R W A,Hinton K.et al, "Green Cloud Computing:Balancing Energy in Processing Storage,and Transport[J]",Proceedings of the IEEE,2011,99(1):149-167.

[6] Younge A J,von Laszewski G,Wang LI-zhe,et al.Efficient Resource management for Cloud computing environmentsw[A],Proceedings of the International Conference on Green Computing,2010[C].Washington,DC,USA,IEEE Computer Society,2010:357-364.

[7] Jeffrey Dean, Sanjay Ghemawat Distributed Programming with MapReduee Beautiful Code,2007,Chapter23.

[8] Wang Z,Tang K,Yao X, "A Memetic Algorithm for Multi-Level Redun dancy Allocation[J]," IEEE Transactions on Reliability,2010,59(4):754-65.

[9] Wang Z,Tang K,Yao X, "Multi-Objective Approaches to Optimal Testing Resource Allocation in Modular Software Systems[J]," IEEE Transactions on Reliability,2010,59(3):563-75.

[10] Qiu J, Lin Z, Tang C, et al, "Discovering Organizational Structure In Dynamic Social Network," 2009[C],IEEE.