Software System Design-Architecture
Assignment 3
C4 System Architecture Design

4 people

November 3, 2019

1 Deploying ADD method in C4 Design

1.1 Important Non-functional Requirements

The important non-functional requirements we identified are listed bellow:

- Availability
- Performance
- Modification
- Scalability
- Interoperability
- Consistancy
- Integrity
- Usability

The constrains we identified are listed bellow:

- No persistent data caching on the agent workstations to limit the implications of local failures.
- No DBs at office locations.
- No administrators at local offices.
- No maintenance down-time.
- The middle layer server cluster tuned for performance.
- The back end tuned for DB performance.
- Well engineered operations architecture.
- High availability cannot be achieved by utilizing fault-tolerant hardware (this option is not economically viable).

The scenarios are listed as following:

Portion of Scenario	Possible Values
Source	End users and internal systems.
Stimulus	Failure and fault: Service off-line, error, crash ans so on.
Artifact	Back up, spare, communication channels.
Environment	Runtime, startup and shutdown, in-service.
Response	Able to response for many requests at the same time. Detect the fault. Recover from the fault. Prevent the fault.
Response Measure	Availability percentage(e.g. 99%) Time to be back to service after an error occured Time to detect a failure

Table 1: Availability Scenario

Portion of Scenario	Possible Values
Source	End users and internal systems.
Stimulus	Many requests come at the same time.
Artifact	A waiting queue or something could caching the request temporarily. A mechanism to serve the request after a short interval.
Environment	In-service.
Response	The C4 should be able to deal with a large bunchs of requests. And after a short interval, the request would be processed.
Response Measure	The time taken to serve every request. The lateness of the processing.

Table 2: Performance Scenario

1.2 Records of ADD iterations

2 Final Software Architecture Documentation

2.1 Documentation Roadmap

2.1.1 Scope and summary

This documentation is built for presence, explanation and analysis of the architecture of Call Center Customer Care(C4) System, which will be employed by ** US telecommunication company. In this documentation, expression and illustration of modules and theirs relationship will be covered, but not all functions of the system are included.

Portion of Scenario	Possible Values
Source	Developers.
Stimulus	Add some interface to fit for a new requirement. Need to improve the system to cater for the increasing demand resulted from the rapid growing users.
Artifact	Code, interface, components and so on.
Environment	Runtime or off-line, in the design process or in the maintaining process
Response	Make the modifications.
Response Measure	The effort, time and money taken to implement such requirement.

Table 3: Modification Scenario

2.1.2 How the documentation is organized

cateloge

2.1.3 View overview

4 Views are employed to illustrate the architecture, including:
Decomposition view: The elements of this view are static modules, and connections illustrate their relationship.
Shared-data view: We using this view to express how important data are shared and protected from inconsistency resulted by business events....
Deployment view: This view also illustrate different parts of the software. Distinguished with module view, it focuses on the runtime status rather than the static status of the system.
All of the three views are following the standard UML specification.

2.1.4 How stakeholders can use this documentation

Use for specification, evaluation, development, test, deployment.

2.2 How a View Is Documented

Refer to the view template

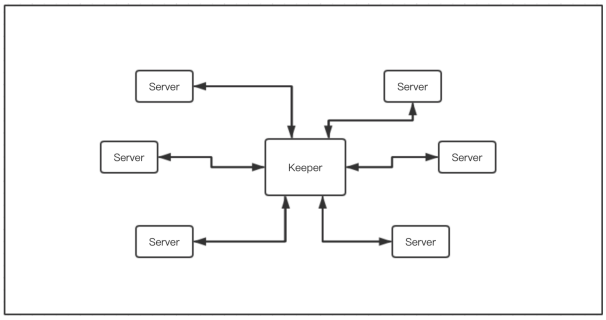
2.3 System Overview

System functions, users, important background, constrains.

2.4 Views

2.4.1 Shared-Data View

Section 1: The Primary Presentation



Section 2: The Element Catalog

Elements and their properties

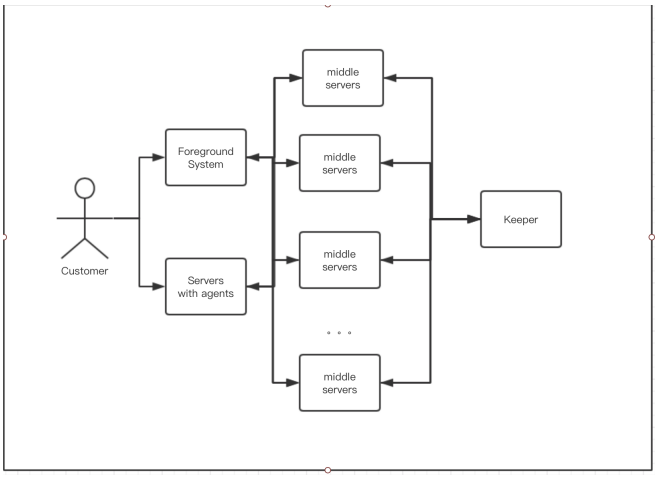
- Keeper The keeper is a data storage center which holds the shared data.
- Server The server is an entity which stores and fetches data from the Keeper. For example, when a customer temporarily terminates the process, server should save the context for a future reference.

Relations and their properties During the fetching process, there may not exist the record, so there should be an exception. Also, in the saving process, to those records that already existed, saving process should be an update to the old version.

Element interfaces The Keeper should provide the find and insert interfaces to the servers.

Element behavior The servers save and fetch the data from the Keeper.

Section 3: Context Diagram

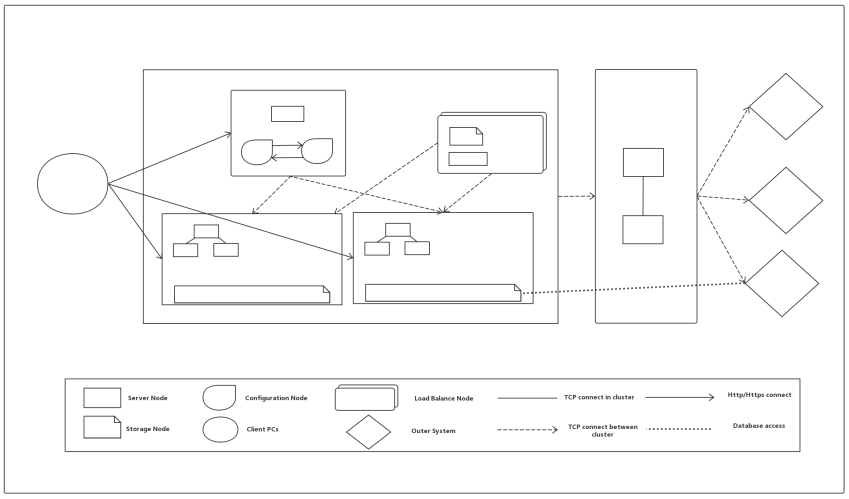


Section 4: Variability Guide Because the data storage is not the only responsibility keeper takes, so in the future, it may be divided into several more specific components, that is to say, the keeper in this view may be changed to a data repository or other data center.

Section 5: Rationale The design problem came from the requirement "A customer can be interrupted (for technical reasons, for example) or suspended by the customer or the representative ... In any case, C4 has to manage the context that persists and can be recalled." In order to do this, there should be a mechanism that stores and fetches the context. There are several options. For example, we can save the information in the agent's PC, or save in the DB provided by the third party. But both of them are infeasible. There is no persistent data caching on the agent workstations, and DB may not be changed since it is provided by the third party. So finally, we choose to save the information in the Keeper, as it is also used to synchronize the event and resolve the conflict as mentioned in the sections before.

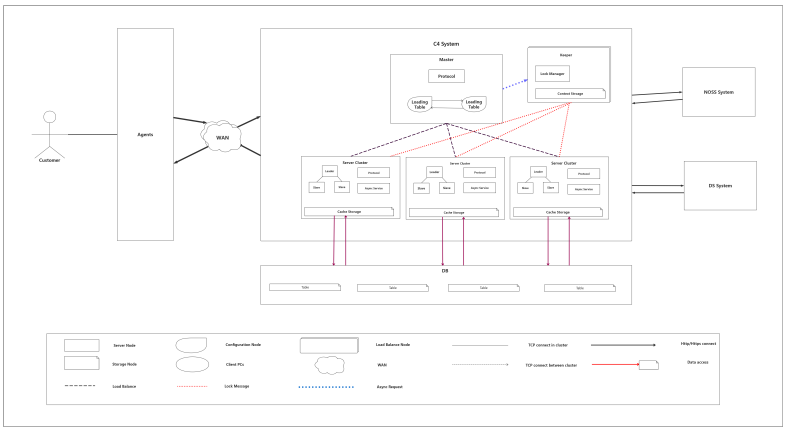
2.4.2 Decomposition View

Section 1: The Primary Presentation



Section 2: The Element Catalog The C4 system is consisted of these important sub-module to decompose: master node , normal server node , storage node , configuration node , load balance node. To resolve the normal phone request (contains the sync-request and async-request) , the normal server nodes are responsible to tackle the service from customers. Moreover , the storage nodes have to save the current session , cache the query on DB , and other cache to the locks . Load balance node has to detect the load in the whole system , and reschedule the load accordingly.

Section 3: Context Diagram



Section 4: Variability Guide The decomposition view represents the template module in the C4 system , and to exercise the variation points , you are

recommended to run the server node. With the sync-request (like fetching user basic information , setting up a new session) and async-request (conflict between the different sessions) , you could catch the key points in the master.

Section 5: Rationale The PC agents communicate with C4 system across the WAN. Master node stands for the whole manager of C4 system . That is , the master node uses the loading table to temporarily record the load status of all subordinate slave server clusters, and can timely alarm and resource reallocation. The master node also contains a protocol subnode, which is mainly used to help the master node process user requests. After the agents send the request, they will first be parsed by the master node to inform the agents of the required service node location, and then the agents will send the request to the target server. The keeper is responsible for asynchronous event processing and session saving between server nodes. Use Lock Manager to allocate locks and save contexts through context storage. The server node is mainly responsible for periodically reporting heart beat status to the master node, and can perform hot backup between nodes. Please note that one server node here is also a cluster. In our design, one cluster is responsible for a certain range. The service (similar to the microservices architecture), and the leader is elected by the vote inside the cluster to perform resource scheduling and service allocation to the subordinate slave. In addition, the server node also supports the processing of the protocol and async service. Responsible for the interaction of the NOSS and DS systems. The cache storage is responsible for the caching of the external DB system. Since each business request will design a query of more than a dozen data tables, considering the time loss of the cascaded query, we use the cache mechanism to The query result is cached inside the server.

2.5 Mapping Between Views

TODO

2.6 Rationale

Explain what decision we have made in our views.

2.7 Directory

3 Personal Remarks

3.1 Statement of ...

3.2 Statement of ...

3.3 Statement of ...

3.4 Statement of ...