**Gideon Server Architecture**

|  |  |  |  |
| --- | --- | --- | --- |
| **리비전** | **날짜** | **작성자** | **내용** |
| 0.1 | 2010-12-06 | 김병수 | 초안 작성 |

목차

[**1.** **목적** 1](#_Toc256070395)

[**2.** **기능** 1](#_Toc256070396)

[**3.** **구조** 1](#_Toc256070397)

[**5.** **구입 대 개발 결정** 4](#_Toc256070398)

[**6.** **구요 데이터 구조 및 알고리즘** 4](#_Toc256070399)

[**7.** **개발 및 서비스 환경** 4](#_Toc256070400)

[**8.** **확장성** 4](#_Toc256070401)

[**9.** **보안** 5](#_Toc256070402)

[**10.** **장애 대체** 5](#_Toc256070403)

[**11.** **위험 요소** 5](#_Toc256070404)

[12. TBD 5](#_Toc256070405)

[13. 메모 5](#_Toc256070406)

1. **목적**
   1. 목표: 안정성 > 단순성 > 유연성 > 성능
      1. 최대한 안정적으로 돌아가야 한다.
      2. 구조 및 코드는 최대한 단순해야 한다.
      3. 적절히 유연하게 하여 변화에 대처할 수 있어야 한다.
      4. 적절한 반응 속도로 동작해야 한다.
   2. 원활한 게임 플레이가 가능해야 한다.
      1. 응답 시간을 최대한 줄여야 한다.
2. **기능**
3. **구조**



그림 1 – 서버(군) 구조도



**그림 2 서버 구동 순서**

* 1. Public Game Servers
     1. Gateway Server
        1. 클라이언트의 연결을 유지하고 multi/broad casting 역할을 한다.
        2. 클라이언트가 로그인하지 않으면 Private Game Servers에의 접근을 거부한다(강제 접속 해제).
        3. 동종 서버 간에는 연결을 하지 않는다.
  2. Private Game Servers
     1. Login Server
        1. 로그인을 위한 인증서를 발급한다.
        2. 클라이언트에게 Zone Server를 배분한다.
        3. 현재 접속자 목록을 유지하여 중복 로그인을 방지한다.
        4. 로그인 성공시 사용자 정보를 제공한다.
        5. 인증서 발급시 유효 시간 및 해쉬 값을 추가하여 인증서 도용으로 인한 로그인을 방지한다.
        6. 전체 서버군이 공통으로 사용한다(여러 서버군에 동시에 로그인할 수 없다).
     2. Zone Server
        1. 하나의 월드를 담당한다.
        2. 클라이언트의 연결을 유지하고 multi/broad casting 역할을 한다.
     3. Community Server(미정)
        1. 채팅
           + 전체 채팅
           + 길드 채팅
           + 범위 채팅
        2. 친구 관리
           + 친구 등록/삭제, 대화
           + 초대, 따라가기
        3. 길드(Company) 관리
        4. 기능에 따라 별도의 서버로 분리할 수 있다.
     4. NPC Server(미정)
        1. Zone Server에서 NPC AI를 분리하여 부하를 줄인다.
        2. 해당 존에 대한 기본적인 정보를 유지해야 하는 문제점이 존재한다.
        3. TODO: 사용 여부를 결정해야 함.
        4. TODO: AI, 길찾기
     5. Market Server
        1. 경매 처리를 담당하여 DB 부하를 줄인다.
  3. Database Servers
     1. DBMS(Database Server)
        1. 직접적인(실시간 동기 호출) DB 쿼리를 원칙으로 한다.
        2. DB Partition을 적극 활용하여 테이블 집중화를 완화한다.
        3. Replication 등의 방법을 이용하여 부하를 분산시킬 수 있다.
        4. TODO: 접속 해제 시의 장애 처리
     2. Database Proxy Server(Cache + Log)
        1. 모든 게임 서버는 Database Proxy Server를 통해 DB를 이용하도록 수정
        2. DB 부하가 집중되는 데이터(계정, 캐릭터 정보 등)에 한하여 캐싱(caching)하여 DB 접근 빈도를 줄인다.
        3. 공유 메모리와 공유 메모리 유지 서버를 통해 장애 복구가 가능하도록 할 수 있다.
        4. 작업 Thread를 허용 범위 내에서 최대한 많이 생성하여 DB I/O blocking으로 인한 지연을 완화한다.
        5. 실시간 처리가 필요하지 않은 쿼리(Insert/Update)를 큐잉(queuing)하여 처리한다.

# **데이터 흐름**

**그림 3 클라이언트 로그인 순서**

1. **변화에 대한 전략**
   1. Evolutionary(Incremental & Iterative) Development
   2. 개발 주기를 짧게 하여 변화에 대응한다(2주~3주).
   3. TDD(Test-Driven Development)를 통해 변화를 쉽게 반영하도록 한다.
   4. 개발 진행 상황의 가시성을 높여 누구나 쉽게 파악할 수 있도록 한다.
   5. 가능한, 데이터 기반(data-driven), 스크립팅(scripting) 시스템을 구축하여 변화에 유연하게 대체하고 생산성을 높인다.
2. **구입 대 개발 결정**
   1. 검증된 Open Source Library(Framework)를 최대한 이용한다.
3. SRPC(<http://srpc.googlecode.com/>) – 기반 라이브러리
4. ACE Framework(<http://www.cs.wustl.edu/~schmidt/ACE.html>) - 기반 라이브러리
5. Boost Library(<http://boost.org/>) - 유틸리티 라이브러리
6. ActiveX Data Object
7. XmlRpc++(<http://xmlrpcpp.sourceforge.net/>) – XML-RPC C++ 라이브러리
8. XSD C++/Tree(<http://www.codesynthesis.com/products/xsd/c++/tree/>) – XML data binding 라이브러리
9. TinyXML(<http://www.grinninglizard.com/tinyxml/>) – XML 파싱 라이브러리
10. Google Test(<http://code.google.com/p/googletest/>) - 유닛 테스트 프레임웍
11. Google Mock(<http://code.google.com/p/googlemock/>) - 유닛 테스트 프레임웍
12. Memcached(<http://www.danga.com/memcached/>) – 분산 메모리 캐시 시스템
    1. 필요할 경우 상용 라이브러리를 구입해 사용한다.
       1. TBB(<http://www.threadingbuildingblocks.org/>) – 코드 병렬화 및 메모리 할당 최적화 라이브러리
    2. 개발 도구
       1. LiquiBase(<http://www.liquibase.org/>) – Database Refactoring Tool
13. **구요 데이터 구조 및 알고리즘**
14. 메모리 풀(memory pool)을 적극 사용하여 성능 및 안정성을 높인다.
    1. 클라이언트와의 패킷은 비트 패킹(압축)을 사용하여 대역폭을 줄인다.
15. Layer pattern: 상위 레벨을 하위 레벨과 분리하여 구조를 단순화한다.
16. Proactor, Reactor, Service Configuration pattern: ACE 기반 패턴.
17. TODO: 좌표 체계, AI, Database schema
18. **개발 및 서비스 환경**
    1. 플랫폼: Windows 200X Server(x64 or x86)
    2. 데이터베이스: MS SQL Server 200X(x64 or x86)
    3. 컴파일러: Visual C++ 9.0
    4. 스크립트 언어: (TODO: ??)
19. **확장성**
20. **보안**
    1. Login Server에서 발급된 인증서를 위조하여 우회 로그인 하는 것을 방지해야 한다.
21. **장애 대체**
22. **위험 요소**
23. TBD
    1. 로그인 후 서버 선택하는 경우와 같이 서버군이 같은 로그인 서버를 공유하는 경우의 처리?
       1. 인증서 개념 도입
    2. Zone Server
       1. 좌표계(공간 분할 방법)?
       2. 사용자가 Zone Server 간에 이동하는 경우의 처리?
       3. NPC Server 분리?
24. 메모
    1. 응답 시간을 최대한 줄여야 한다.
    2. 서버간 연결(S2S)을 다중으로 하여 성능을 높인다(connection pooling).
25. …