게임 서버 종류

* 로비서버

ex)배틀넷

채팅 서버의 확장

* MMOG서버

MUD의 서버가 효시

서버 부하가 큼.

"이번 학기 강의의 주 관심사."

게임 서버의 설계 목표(안정성, 서능)

* 쾌적한 게임 환경
* 동접
* 핫스팟

성능 향상을 위한 해결책

* 효율적인 프로그래밍
* 부하 분산
  + 서버 분리(샤딩)
    - 월드 복사 & 분리
    - 다른 월드로 이동 불가(돈내면 이전 해줌, DB분리)
    - 통합하는 추세(경매장 시스템)
  + 기능별 분산
    - 사용자 분리 없이 확장
    - 부하가 커 여러 종류 서버 조합으로 구성 (Main, Login, AI, DB, Query 등)
    - 한계
      * Main Server
        + 부하가 집중됨.
        + 더 이상 기능적으로 나눌 수 없음(통신 부담 증가)
    - Main Server의 부하 감소를 위한 분할
      * 공간 분할 Zone 서버
        + 전체 맵을 지역별로 나누어서 다른 서버가 관리
        + Portal, Seamless
      * 접속 분할 Duplicate 서버
        + 샤딩과는 다름.
        + 동일한 공간의 Clone을 만들어서 각각 다른 서버가 관리.
  + 공간 분할(월드 분할 ; 맵?)
    - Seamless
      * 경계선 근처에 서있는 물체를 2개 이상의 서버가 동시에 관찰 할 수 있어야 함
      * 구현하기 어려움
    - Portal
      * 게임성이 떨어짐.(로딩, 플레이의 단절)
      * 구현하기 쉬움
      * 인던이 여기에 속함
  + 정적 분할과 동적 분할

Zone Server

Zone Server 
master (FrontEnd HA): 
zone : 34 Ell E CI-CF 
Client 
Master 
Server 
Zone 
Server 
Zone 
Server 
Zone 
Server 
Main Server 

Duplicate Server

Duplicate Server 
Client 
Main 
Server 
Main 
Server 
Main 
Server 
DB 
Server 

##############

게임서버를 위한 하드웨어와 소프트웨어

**하드웨어**

하드웨어 지식이 필요한 이유?

* 서버에 걸리는 부하가 많기 때문에 하드웨어의 성능을 최대로 이끌어내는 프로그래밍 필요
* 성능을 최대한 끌어내려면 하드웨어가 어떻게 성능에 영향을 미치는지 파악

중요 하드웨어

* CPU
* Memory
* Network

CPU

* X86계열이 대세 (ARM은 사용불가.)
* AMD, Intel 2가지 계열이 있다.
  + 서버용 CPU가 따로 존재한다.(Intel Xeon, AMD EPYC)
* 64bit, Multi-core
* 속도와 캐시 크기, 메모리 Bus대역폭이 중요하다.
* 서버용, 일반 데탑용 CPU 차이
  + 서버용은 Multi Processor를 지원한다.
    - 캐시 동기화 지원
  + 서버용은 서로가느이 데이터 전송 전용 통로를 제공한다.
    - 암드 : 하이퍼트랜스포트
    - 인텔 : QPI

СРИ 
NOW 
СРИ 
СРИ 
CPU 
СРИ 
Метогу 
СРИ 
СРИ 
Метогу 
Метогу 
CPU 
CPU 
Метогу 
Memory 
Controller 
Метогу 
OLD 
СР ИД o_b0bXl 
ДбО11 

64bit 이슈

ㆍ 64비트 이슈 
64 비트를 사용한다. 
OS와 Compiler도 64비트 버전을 사용해야 한다. 
기존의 32비트 CPU로는 최대 4GB의 메모리 
밖에 쓸 수 없었기 때문에 서버 용량에 제한이 
많았었다. 
64 비트의 경우 16ExaByte의 메모리가 
가능하다. 
프로그래밍 시 Int type과 Pointer type의 
크기가 달라지는 것을 주의 해야 한다. 

Multi-processor

* SMP
* 빠른 네트워크 응답속도와 처리 속도 개선을 위해 Multi-processor를 사용

Multi-core

* 발열에 막힌 CPU의 성능향상 제한을 극복하기 위한 궁여지책
* 기존의 4~8개의 CPU를 활용하던 프로그램 방식에서 8~64개의 코어를 활용하도록 변경필요.
* 앞으로도 코어 갯수가 늘어날 예정

둘의 차이점

* SW적으로는 차이 X
* HW적으로는 메모리 접근시 성능 차이 존재
  + CPU의 개수만큼 메모리 대역폭 증가
  + NUMA 문제

CPU 발전의 트랜드

* 클럭속도 증가
  + 반도체 생산 공정 개선
  + 한계에 부딪힘 : 발열
* 클럭당 수행되는 명령어의 개수 증가
  + 아키텍쳐의 개선
  + 파이프라인, 슈퍼스칼라, 슈퍼파이프라인, Out-of-order
  + 캐시 용량 증가
  + 한계에 부딪힘 : 한계효용의 법칙
* Core 개수 증가
  + 현재의 방식, 프로그램 작성 방식이 바뀌어야 함
  + 발열로 인한 클럭속도 향상의 한계가 주 원인
  + 현재 32-Core까지 (앞으로 계속 늘어날 예정)

파이프라인의 발전

* "어떤 명령어를 수행하는데 몇 싸이클이 걸리는가?"라는 질문은 의미가 없음
* 파이프라인이 무효화 되지 않는 한 프로그램의 실행속도는 메모리 Read에 종속됨
  + 명령어가 더 많아도 메모리 Read가 적으면 더 빠름

SIMD 명령어의 발전

* 하나의 명령어로 여러 개의 실수를 동시에 연산
* MMX -> SSE -> AVX

파이프라인의 고도화에 따른 주의

* 파이프라인을 리셋 시키면 손해가 너무 크다
  + Pentium 4의 경우 31단계
* 리셋의 원인
  + 시스템 콜
  + 분기 예측 오류
  + Interrupt, Exception
* 대책
  + 시스템 콜을 될 수 있으면 하지 말 것
  + If, switch 등을 자제한다.

CACHE

* 프로그램 실행속도에 가장 큰 영향을 미치는 요소
* Cache가 큰 CPU일 수록 속도가 빠르다

tip

* 같이 쓰이게 되는 데이터는 묶어 놓는다.
* 루프 안에서 사용하는 데이터는 캐쉬에 다 올라올 수 있도록 한다.
* int대신에 hallf나 char를 사용한다.

Multi Processor Programming

* 잘하면 N배 성능향상, 못하면 성능하락
* Lock을 줄여라
  + Lock을 실행하는 순간 모든 CPU의 cache사용이 중지됨
  + Semaphore, Condition 변수는 시스템 콜
* Cache Thrashing에 주의 하라
  + Cache는 line단위로 움직임

ㆍ Memory 
서버가 요구하는 용량을 
제공하면 된다. 
일 반적 인 Desktop용 
메모리가 아니라 
Er℃r수정기능이 있는 
특수 메모리를 사용한C 
대역폭이 크지만 반응 
속도는 느리다. 
nMSUN 
SAMSUNG PC25初 Fully Buttered 그 
고성능2| 안정성을 통시ㆍ ! ! 
일시기는 상황만의 기출 들의 
힡을 수유는 세||4 
l,l청여져 시올지 출시한 5]|℃ e니니焉: 다,” IGB 는 
서버 말 일그스티l이선에 墨•칯8|여 다국 매넣! 콤`•하는 2로희 4콤으! 67WzZ! 
1,.• 속•구로 고X\8 한통통혜A|도 빠츠납니Q. 
회나 3.Ä囫" 의 장」 대역矍올 ,|콤하고. 재넓의 *|콤 한4하는 ECC기콤9로 테:월일무ㆍ 
,1는 0니[뙤 

ㆍ Network 
10M -> 100M -> IG -> 10G 까지 발전 
서버간의 연결을 위해 한 서버에 여러 개의 
네트워크 카드를 꽂아서 사용하기도 한다. 
확장 버스의 대역폭을 고려해야 한다. 
PCI (32bit, 33MHz) : 133MB/s 
PCI-Express 2.0 (1 lane) : 500MB/s 
속도가 더 필요하면 Infiniband사용 
Effectiw unid•æctiunal theuæt•cal throughout 
(actual data rate, not signal•ng 
SD닣 
00닣 00나 [Ok-10 
EDR 
1※ 2 Gait'S 4 돈 1.크F4 Gait,s 
IEGI」i" GI」if/s GI」if/s GI」I,s 1」」 
1 2※ 24 16364 

=================================================

**소프트웨어**

서버용 운영체제의 종류

* Unix계열
  + 리눅스, freeBSD, Solaris, Osx
  + 가격이 저렴하다.
  + 유지 보수 관리가 어렵다. (잘 아는 사람이 필요하다)
* Windows 계열
  + Windows 2008, 2012, 2016
  + 비싸다
  + 유지 보수 관리가 비교적 쉽다.(배우기 쉽다. 신경 쓸 것이 적다)

정리 
• 게임서버의 최적화는 HW를 고려해야 한다. 
컴퓨터 구조에서 배운 것을 실제로 고려해야 
한다. 
System Call 하지 말기 
Cache 잘 사용하기 
가능하면 메모리 적게 사용하기 
Multi-Thread Programming (다음달에..) 