Jelaskan apa saja perbedaan antara NFS, AFS, GFS dan Ceph, dengan memperhatikan tujuan pengembangan dan penggunaan sistem tersebut.

Jawaban:

1. **<u>Network File System (NFS)</u>**
   a. **Tujuan Pengembangan**
      - Menyediakan transparansi akses file jaringan untuk sistem UNIX yang disimpan di server jarak jauh
      - Implementasi sederhana berbasis client-server menggunakan RPC
      - Dirancang untuk lingkungan jaringan lokal (LAN) dengan latensi rendah sehingga lebih compatible dan mudah digunakan

   b. **Karakteristik**
      - Arsitektur: Model client-server sederhana
      - Stateless: Tidak menyimpan informasi status client
      - Transparansi: Pengguna dapat mengakses file remote seperti file lokal
      - Caching: Caching di sisi client untuk meningkatkan performa
      - Konsistensi: Validasi timestamp untuk menjaga konsistensi cache
      - Skalabilitas: Terbatas karena arsitektur terpusat

   c. **Penggunaan**
      - Berbagi file dalam jaringan lokal perusahaan
      - Lingkungan dengan jumlah pengguna terbatas
      - Situasi di mana kesederhanaan lebih penting daripada skalabilitas
      - Cocok untuk file berukuran kecil hingga menengah

2. **<u>Andrew File System (AFS)</u>**
   a. **Tujuan Pengembangan**
      - Mengutamakan skalabilitas untuk mendukung ribuan pengguna
      - Dirancang untuk mendukung ribuan pengguna secara bersamaan
      - Mengurangi beban server dengan whole file caching

   b. **Karakteristik**
      - Arsitektur: Menggunakan model client (Venus) dan server (Vice)
      - Caching: Caching seluruh file di sisi client (bukan hanya bagian file)
      - Callback: Mekanisme callback untuk menjaga konsistensi cache
      - Namespace Global: Namespace file yang konsisten di seluruh sistem
      - Stateful: Server menyimpan informasi tentang client yang meng-cache file

   c. **Penggunaan**
      - Lingkungan kampus atau perusahaan besar dengan ribuan pengguna
      - Situasi di mana skalabilitas sangat penting
      - Cocok untuk file yang sering dibaca tetapi jarang diubah
      - Mendukung mobilitas pengguna agar bisa bekerja di berbagai lokasi

3. **Google File System (GFS)**
   a. **Tujuan Pengembangan**
      - Menangani massive amount of data (petabyte scale)
      - Dirancang untuk Google's specific needs (web crawling, indexing)
      - Toleransi tinggi terhadap kegagalan hardware murah
      - Optimasi untuk file besar dan sequential access
   b. **Karakteristik**
      - Arsitektur: Single master dengan banyak chunkserver
      - File Model: File sangat besar (100MB+), write-once, append-mostly
      - Chunk Size: 64MB (sangat besar)
      - Replikasi: 3+ replika per chunk
      - Konsistensi: Relaxed consistency model
   c. **Penggunaan**
      - Aplikasi big data dan analitik
      - Penyimpanan dan pemrosesan file sangat besar (gigabyte hingga petabyte)
      - Lingkungan di mana throughput lebih penting daripada latensi
      - Mendukung model pemrograman MapReduce
      - Basis untuk Hadoop Distributed File System (HDFS)

4. **Ceph**
   a. **Tujuan Pengembangan**
      - Unified storage yang menyediakan object, block, dan file storage dalam satu platform
      - Software-defined storage untuk hardware komoditas
      - Skalabilitas horizontal yang sangat tinggi
      - Eliminasi single point of failure
   b. **Karakteristik**
      - Arsitektur: Fully distributed dengan RADOS sebagai fondasi
      - Komponen: Monitor, Manager, OSD (Object Storage Daemon)
      - Storage Types: Object (RGW), Block (RBD), File System (CephFS)
      - Penempatan Data: CRUSH algorithm untuk distribusi otomatis
      - Durabilitas: Replikasi dan erasure coding
      - Skalabilitas: Horizontal scaling tanpa batas teoritis
   c. **Penggunaan**
      - Cloud storage dan infrastruktur cloud
      - Penyimpanan untuk virtualisasi dan container
      - High-performance computing (HPC)
      - Backup dan archiving skala besar
      - Lingkungan yang membutuhkan fleksibilitas tinggi dalam tipe penyimpanan

Tabel Perbandingan NFS, AFS, GFS, dan Ceph

Aspek	NFS	AFS	GFS	Ceph
Fokus Utama	Kesederhanaan dan transparansi	Skalabilitas dan performa	Throughput dan toleransi kesalahan	Skalabilitas dan fleksibilitas
Caching	Di sebagian file	Di seluruh file	Tidak ada caching	Terdistribusi merata
Konsistensi	Menggunakan Timestamp Validation	Menggunakan callback promises	Menggunakan relaxed consistency	Menggunakan strong consistency
Arsitektur	Client-server sederhana	Client-server hierarkis	Master-slave	Fully Distributed
Skala	Kecil hingga Menengah	Besar	Sangat Besar	Ekstrem Besar
Toleransi Kesalahan	Terbatas	Menengah	Tinggi	Sangat Tinggi
Tipe Workload	General Purpose	Read-Heavy	Append-Heavy	Beragam
Kompleksitas	Rendah	Menengah	Tinggi	Sangat Tinggi