1.

The Case resolver would be better.

When requesting something that is cached, the Case DNS server will offer a much faster response time as the request doesn’t even have to leave the local network. Therefore the Case DNS is the optimal configuration.

2.

Yes this would be worth doing.

The average time of a search would overall drop under 2 seconds.

While 60 percent of the time the search would be 2.05 seconds, 40 percent of the time the search would take .1 seconds. This averages out to around 1.27 seconds which is well under 2 seconds making it a worthwhile upgrade.

3.

a. a- Foo.com 160.50.1.1 as well as CNAME movies.foo.com and cdn.com

b. User V sends a DNS request to server W, a local DNS server

W doesn’t have a cached DNS record, so it sends V’s request to root DNS server X

X then gives W the IP address of .com DNS server Y

W sends V’s request to Y

Y then replies to W that the hostname V requested as authoritative DNS server Z

W sends a query to Z with the IP given from X which is 27.0.0.1 (foo.com/home/)

W returns the IP to V

V then makes its HTTP request (for foo.com/home/) to 27.0.0.1

Foo.com/home then gives V the requested data

c. User V sends a DNS request to server W, a local DNS server

W doesn’t have a cached DNS record, so it sends V’s request to root DNS server X

X then gives W the IP address of .com DNS server Y

W sends V’s request to Y

Y then replies to W that the hostname V requested as authoritative DNS server Z

W sends a query to Z to get the IP that is linked to foo.com/home/

Z replies to W with the info that W really wants cdn.com

W sends a query to Y for the hostname to cdn.com

Y replies with the IP of authoritative DNS server Z2

W sends a query to Z2 asking for the IP linked to cdn.com

Z2 replies with the IP to W

W sends the IP to V

V finally establishes an HTTP connection with cdn.com and request the file topten.html

4.

a. HTTP process

***Request sent:*** inc.com GET /home/cn

***Response:*** HTTP/1.1 200 OK <chinesedata> Set-cookie:1337 (for user ID)

The server then stores that id 1337 now has chinese as its preferred language

***Request sent:*** inc.com GET /womensummit&cookie:1337

***Response:***HTTP/1.1 200 OK <chinesedata>

***Request sent:*** inc.com GET /edusummit&cookie:1337

***Response:***HTTP/1.1 200 OK <chinesedata>

b.

***Request sent:*** inc.com/login

***Response:*** HTTP/1.1 200 OK <logindata> set-cookie:1287 (user id Cookie)

The server links the id 1287 with the entered login information

***Request sent:*** inc.com/memberforum&cookie:1738 (auto login now occurs)

c.

***Request sent:*** inc.com/purchasepage/

***Response:*** HTTP/1.1 200 OK <purchasepagedata> set-cookie:1738 (user id cookie)

The server records the actions of the user, in this case that they viewed the purchasepage

d.

This is done with the inclusion of a file on the partner sites that will record your actions there and report them back to the main site. This is done by forcing you to make an HTTP request to the site even if it isn’t the main site.

Specifically the partner would do set-cookie:1789 (same from main site) to their response, and you would add the cookie. The partner server would then record that you visited it.

5.

a. To maximize utilization send the file to more than one user at a time. Sending the file to 10 users at the same times, and then sequentially selecting clients would allow for the total time to be calculated. It should be the file size times the number of users divided by the bandwidth. So then this is the equation:

5,000MB \* 100 / 1MBps (10 mbps) = 500,000 seconds

b.

Again start by sending the data to 10 of the users. They will all then send the data on to another user until all users are receiving the requested data. Using this strategy the delay is negligible as the packets send right after they are received to the next user. This leads to a theoretical time of:

5,000MB \* 10 (effective) / 1 MBps (10 mbps) = 50,000 seconds.

6.

a. This is possible. You don't need to upload data to receive the files, you just need others doing it as long as enough seeders sending the data out from their machines. This is leeching though, and is normally not very helpful to the system as a whole. If everyone did that the system wouldn't work.

Specifically an unknown number of peers will be unchoked in parallel. Some are selected if they are uploading his content and the fastest uploaders are selected and unchoked for data transfer. This delivers the content.

b. If he has complete control over the machines this is true. He can have each machine leech certain sections of the program. Then they can be combined afterwards on one machine. Whether this is faster or not will depend on the speed of your connection. It may not be faster if you download each chunk on a gigabit connection but then have to combine the file over a 10mb/s connection (no idea why that would happen). So yes, under normal circumstances I would say it’s much faster.