Capacity Estimation

execute typical instruction	1/1,000,000,000 sec = 1 nanosec
fetch from L1 cache memory	0.5 nanosec
branch misprediction	5 nanosec
fetch from L2 cache memory	7 nanosec
Mutex lock/unlock	25 nanosec
fetch from main memory	100 nanosec
send 2K bytes over 1Gbps network	20,000 nanosec
read 1MB sequentially from memory	250,000 nanosec
fetch from new disk location (seek)	8,000,000 nanosec
read 1MB sequentially from disk	20,000,000 nanosec
send packet US to Europe and back	150 milliseconds = 150,000,000 nanosec

Storage Estimation:

Total storage => Number of point_pairs * STORAGE_COST_PER_PAIR

STORAGE_COST_PER_PAIR = cost of storing a pair((f1, t1), (f2, t2))
= (f1 : f2 : delta_time) = 3 integers = 3 * 4 bytes = 12 bytes.

Total storage => Number of point_pairs per song * total number of songs * 12Bytes

Assume the total number of songs in the database to be 1 million.

Total storage => Number of point_pairs per song * 1M * 12Bytes

Total storage => NUMBER_OF_PAIRS_PER_CHUNK * AVERAGE_NUMBER_OF_CHUNKS_PER_SONG * 1M * 12Bytes

AVERAGE_NUMBER_OF_CHUNKS_PER_SONG = avg length of song / divided by length of chunk.

Assume average length to be around 3 minutes = 200, and each chunk to be around 10 seconds.

AVERAGE_NUMBER_OF_CHUNKS_PER_SONG = 200/10 = 20.

Total storage => NUMBER_OF_PAIRS_PER_CHUNK * 20 * 1M * 12Bytes

NUMBER_OF_PAIRS_PER_CHUNK = points * (points - 1) / 2

If there is one interesting point per second, points_per_chunk is
10 seconds * 1 point/second = 10 points.

NUMBER_OF_PAIRS_PER_CHUNK = 10 * (10 - 1) / 2 ~=50

Total storage => 50 * 20 * 1M * 12Bytes => **12 GB**

Taking into account replication for performance and fault tolerance, we multiply by a factor of 3, to get a storage requirement of **36 GB**.

Processing estimation:

TOTAL_PROCESSING_TIME_PER_SECOND = Requests * number of pairs in clip * SEARCH_TIME

Assume number of requests to be 1000 per second. For a clip of 10 second length, we can assume 10 interesting points. Thats 10 * (10 - 1) / 2 ~= 50 pairs.

TOTAL_PROCESSING_TIME_PER_SECOND = 1000 * 50 * SEARCH_TIME

SEARCH_TIME = time to search (fx : fy : delta_time_xy) in the DB.

We can hash each pair in the database and create an index on it. Take each pair from the clip and search against PAIR_ID.
PAIR_ID is the hash of a pair in the database.

The chunk having the maximum number of matches with the request pairs is the answer. This can be found with an inverse mapping of PAIR_ID -> CHUNK_ID.

We have about 1 billion pairs in the DB. With sharding on PAIR_ID, the time required to match an incoming pair will be ~5ms.

TOTAL_PROCESSING_TIME_PER_SECOND = 1000 * 50 * 5 ms
= **250 seconds/second**

To meet our requirements, we need at least 250 processing units. To keep server load around 60-70%, we should have 250 / 0.6 ~= **400 processors**.