Calculating data storage:

Let's assume we are storing the following for simplicity: IP address, geographical location of the user, time of visit, URL of visited page, unique userID (auto-generated from our system). If we convert the values to string, IP - 15 characters, geographical location (eg. 35.96689,-95.929945) - 22 chars, time of visit (format of YYYYMMDD hh:mm:ss) - 16 chars, URL - 100 chars, userID - 32 chars; total of roughly 200 chars.

Let's assume 10 billion write events per day - 10 billion \* 200 chars = 2 trillion ~ 4 terabytes if assuming 2 bytes (16 bits) per character (no compression) per day.

If data storage is not in cloud, utilize SSD hard-drives to expedite the read/write access.

Database technology:

We’ll use PostgreSQL for storing data. It supports parallel (multi-core) queries & materialized views. Cstore\_fdw (by CitusDB) which is an open source columnar store extension for PostgreSQL which has notable benefits for analytics use cases where data is loaded in batches. We should regularly index tables for faster write/retrieval. We should also shard DB to spread data across different machines to help writing data.

Metrics for clients:

We can cache the data in system (probably dedicated servers for caches) with memcached (open-source). It can save lots of reads directly from DB and provides faster & efficient way to provide metrics to customers – even real-time values (only for recent activities).

Data processing:

Use Apache Storm – super low latency and near real-time processing (whereas Hadoop is slow and need to be computed behind the scene). Process calculations such as total visits per page, average time spent on page, # of visits per city/country are executed. Every new data (user interactions) should be calculated on top of previous values (very similar to moving average coding question logic). If used with Trident, it gives us option to use micro-batches (instead of pure stream processing) – this can be used to reprocess historical data in case of bugs in the processing logic (also can use moving average logic to calculate last N elements, or interval of values).

Load-balancing:

Need very reliable distributed coordination, low latency, and high throughput – we can use Apache ZooKeeper which can be easily integrated with Storm.

Downtime:

When resharding or upgrading DB, there could be some period of downtime. Citus Cloud 2 boasts about zero downtime but it’s costly. Resharding should be thoroughly investigated to minimize downtime – potentially only allowing read-only access during few seconds/minutes of resharding.