"person holding red cricket bat near seashore" by Lochie Blanch on Unsplash

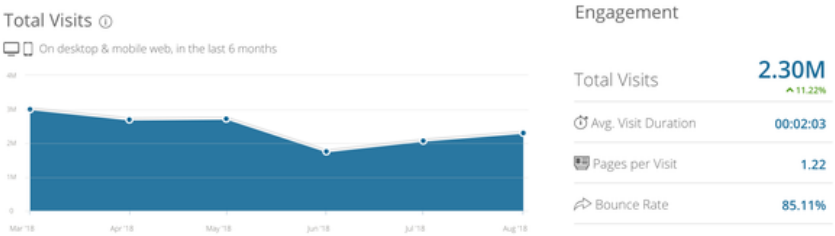
CricInfo / CricBuzz System design

Narendra L

Sep 29, 2018 · 5 min read

TRAFFIC: Before designing a system design for CRICINFO we need to know the traffic. you can assume or go to

https://www.similarweb.com/websi... or Alexa Website Traffic, Statistics and Analytics

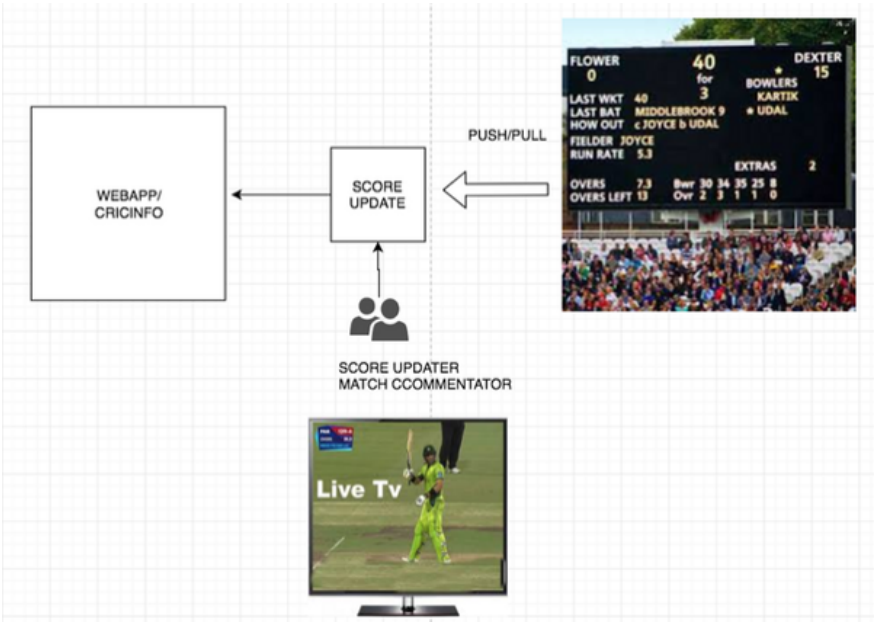


so approximately 2.5M visits every day.

Now you know where your data centers should be placed!! and where your user base is!!

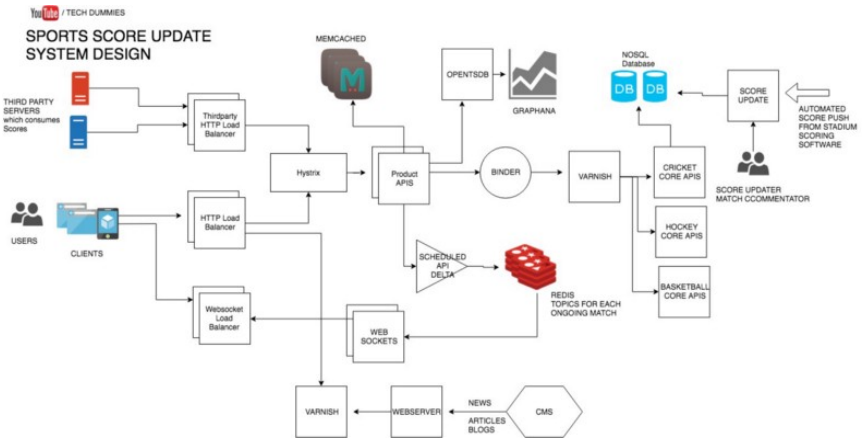
If you are too lazy to read watch this video

How do scores are imported to Cricinfo's system?



Its a combination of automatic and manual.

1. There is a scorer and a commentator, often in the same cabin, who input that data. Usually, the scorer inputs the runs scored off the ball, whether any extras were conceded or not, where the ball was played (on the ground) etc. The scorer also inputs the next batsman when he walks in, the mode of dismissal, the bowler who comes into bowl
2. Or the scores are imported directly from Stadium scoreboard software by push or pull or socket.



Product APIS

Cricinfo/Cricbuzz websites not only consume data for their sites, but they also feed data to vendors/Third parties like apps like Games, other

sports websites and TV channels etc.

What are the information which they sell/share?

- Athletes: Biographical, profile, and statistical data for sports' biggest stars, covering all major sports
- LIVE SCORES
- Calendar
- headlines
- leaderboards
- Teams
- photos etc

As these APIs are paid the service provider should prioritize the traffic and also sometimes customizable and high availability.

Scalable APIs :

There are so many vendors, everyone has different/custom feeds/API requirements but writing those many APIs is the case. So we need to have generic API s

Cricinfo has two kinds of APIS

1. Core APIS
2. Product API

Most of the time a single product API is a combination of many core APIs, in that case, Binder will call the core APIs in parallel and gets the product API ready.

Binder is a component which handles calling core APIs (asynchronously) and merging and returning to product API.

What if one of core API call fails while calling Asynchronously?

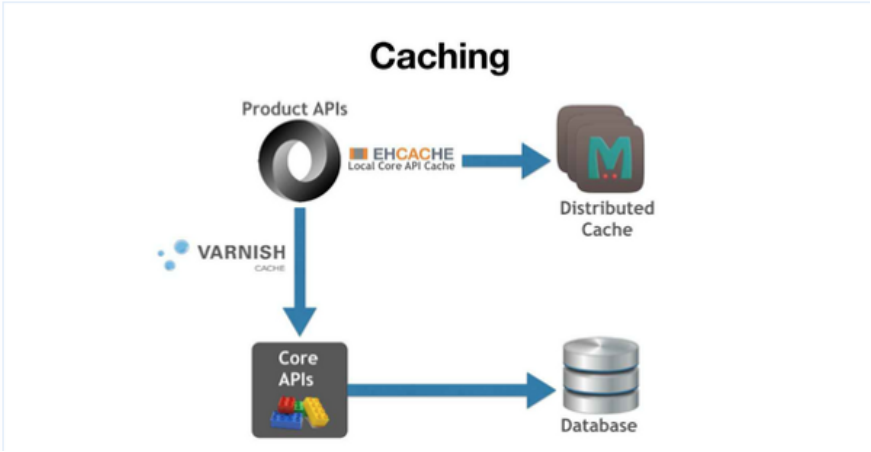
Make the main call/Product API call also fail !!

Why? To have predictable output. as we can't send incomplete data to third-party as third-party consumers might have not handled the scenario of incomplete/partial data

Content search engine:

Used to sort the different content like Editorial content, WordPress content, scores, highlights etc based on whats most relevant based on the past 24 hours. which returns the relevant content to show to users.

CACHING:



Cricinfo uses 3 tiers caching!

1. Local cache to save hot/famous APIs and its local because to avoid network calls to distributed cache(EH cache)

2. If the data is not found in the Local cache then go to Distributed cache(Memcache)

3. Varnish between the binder and Core APIS to avoid stampede effect. eg: when about 30 to 40 calls for the same API are coming to varnish, it combines all of that API calls and get it once and cache it for few seconds also

This stampede can be avoided by Request collapse : serve response to all similar request in a few millisecond window so everyone gets the response and once this response is cached in Varnish.

Hystrix:

Hystrix is a latency and fault tolerance library designed to isolate points of access to remote systems, services and 3rd party libraries, stop cascading failure and enable resilience in complex distributed systems where failure is inevitable.

It is used to do Circuit breaking for all of the Microservices(Core apis and product APIs) used.

REAL-TIME DATA UPDATE USING REST APIS!!

As sports fans need data in real-time there should be a way to push data to clients

Cricinfo has capabilities to send 1 Million WebSocket Updates to Fans in less than 100MS.

How do they do it? they use Redis queues which are mapped to each match which is happening at day, any updates related to that match will be posted to that queue.

1. A scheduler will keep on querying the scoring and match statistics API
   (Poll couple of times(2–3 times) every second to check if there is change in the API data, if there is then send it to next stage)
2. Then the data will be sent to check the difference of the data sent previously VS current to reduce the payload size. only the new information will be sent to respective Queue
3. WebSocket connection from the client to WebSocket server FANOUT data which is Redis queue to clients.

For old games which already happened you never need to use Websockets as the data is readliy available and can be queried via APIS

How to do API monitoring

OPENTSDB: Scalable Time Series Database Store and serve massive amounts of time series data without losing granularity.

- Runs on Hadoop and HBase
- Scales to millions of writes per second
- Add capacity by adding nodes
- Generate graphs from the GUI
- Pull from the HTTP API
- Supports Graphana for visualization of data

Graphana:

Grafana allows you to query, visualize, alert on and understand your metrics no matter where they are stored. Create, explore, and share dashboards with your team and foster a data-driven culture.

Databases:

MongoDB

Couch DB

Cassandra

Riak

Membase

Redis