Summary assessment from user's perspective

From the perspective of the user, this site does not have upfront usability. The interface is clunky and does not inspire the user to continue with the service, nor is it well documented what each page should mean to the user and how to navigate between pages. However, for the simplest analytics engine, the user is able to move between pages and view the relevant information about each site that was promised in the product expectations. As far as the user is concerned, he is looking at a quick and unpolished project that took a few hours to get up and running, but manages to deliver the requested functionality.

Summary assessment from developer's perspective

From the perspective of the developer (me), this project was an exercise in learning the basics of the Ruby programming language and the Rails platform. The design was supposed to be minimalistic in order to reduce the complexity necessary to have a working implementation. As such, the user interfaces are simple HTML and there are not instructions embedded on each page. The backend was the focus, and the backend works to spec.

Most and least successful decisions

Because of the learning nature of this project, and the evolving requirements through each stage, the most successful design decisions were those that did not depend on the step of the project. This means that the object model with Sites -> ID -> Hit Count as the core was a good design decision; this model stood up to each iteration of the final product. The least successful decision was regarding the Pageview object, which currently belongs to Site. The original design was to avoid the need for a Pageview table by recording hits and durations separately, thus reducing the DB load when sites are being slashdotted. However, in order to implement client location at the proper granularity, a Pageview table needed to be created. I still believe this to be a poor design decision because of the load that will be necessitated on the analytics server and databases.

Analysis of design faults in terms of design principles

The authentication system is not a design fault, but more of an implementation hole. The design of the authentication is rational: each user has one email and password, and no two users may have the same email. However, the implementation does not prevent users from "claiming" the same email address and provides no helpful information when site registration fails. The design faults related to the Pageview table and the Pages table come clear with the load required of the analytics server. Each request denoting a hit to a site causes a row insertion in the database. This means that load scales linearly with the aggregate load of client sites, and is clearly an unsustainable model. A malicious user is able to cause a DOS attack on the analytics site by simply spoofing a Slashdot effect on another client. The tables are also unrecoverable in the event of catastrophe, so there is little fault tolerance.

Priorities for improvement

First priority for improvement is certainly the authentication clarity and security. Following that would be usability improvements to make navigation and understanding a little easier. Finally the Pageview table could be extracted out of the Site object, or removed completely (a redesign of the records table would be necessary).