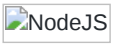
[Dolphin] NodeJs,SocketIO and MongoDB

NguyenSiTrung edited this page on Oct 23, 2016 · 7 revisions



A.Tổng quan về NodeJS

NodeJS là gì?

Node.js là một mã nguồn được xây dựng dựa trên nền tảng Javascript V8 Engine. Node.js được phát triển bởi Ryan Dahl vào năm 2009 và phiên bản mới nhất của nó là v0.10.36. Định nghĩa của Node.js như được cung cấp bởi các tài liệu chính thức của nó là như sau:

• Node.js là một nền tảng được xây dựng trên thời gian chạy JavaScript của Chrome để dễ dàng xây dựng các ứng dụng mạng nhanh và khả năng mở rộng. Node.js sử dụng một hướng sự kiện, non-blocking I / O mô hình hướng sự kiện mà làm cho nó nhẹ và hiệu quả, hoàn hảo cho các ứng dụng thời gian thực dữ liệu chuyên sâu mà chạy trên các thiết bị phân phối.

• Node.js là một mã nguồn mở, cross-nền tảng môi trường thời gian chạy cho phát triển phía máy chủ và các ứng dụng mạng. Ứng dụng Node.js được viết bằng JavaScript, và có thể chạy trong thời gian chạy Node.js trên OS X, Microsoft Windows, và Linux.

• Node.js cũng cung cấp một thư viện phong phú của các module JavaScript khác nhau mà đơn giản hóa việc phát triển các ứng dụng web sử dụng Node.js đến một mức độ lớn.

Đặc điểm của Node.js

Sau đây là 1 số tính năng quan trọng mà làm Node.js sự lựa chọn đầu tiên của kiến trúc sư phần mềm:

• Không đồng bộ và và phát sinh sự kiện trong quá trình sử dụng. Điều đó có nghĩa là tất cả các API của thư viện Node.js là không đồng bộ nghĩa là không bị chặn. Nó chủ yếu có nghĩa là một máy chủ dựa trên Node.js không bao giờ chờ đợi cho một API để trả lại dữ liệu. Việc di chuyển máy chủ đến các API tiếp theo sau khi gọi nó và một cơ chế thông báo các sự kiện của Node.js giúp máy chủ để có được một phản ứng từ các cuộc gọi API trước.

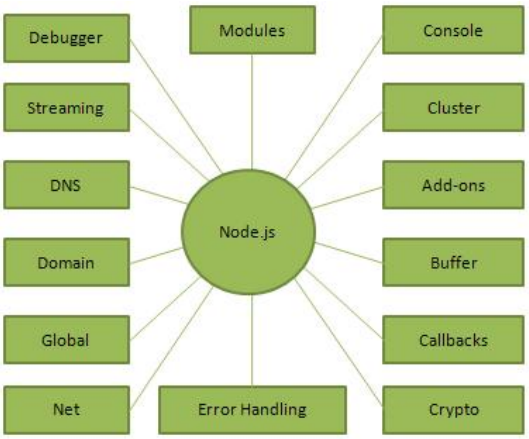
• Tốc độ thực thi chương trình rất nhanh: Được xây dựng trên cơ V8 JavaScript Engine của Google Chrome, thư viện Node.js là rất nhanh trong thực thi mã.

• Độc Threaded nhưng cao Scalable - Node.js sử dụng một mô hình luồng duy nhất với sự kiện lặp. Cơ chế tổ chức sự kiện giúp các máy chủ để đáp ứng một cách không ngăn chặn và làm cho máy chủ cao khả năng mở rộng như trái ngược với các máy chủ truyền thống mà tạo để hạn chế để xử lý yêu cầu. Node.js sử dụng một chương trình đơn luồng và các chương trình tương tự có thể cung cấp dịch vụ cho một số lượng lớn hơn nhiều so với yêu cầu máy chủ truyền thống như Apache HTTP Server.

• Không Buffering - Node.js ứng dụng không bao giờ đệm bất kỳ dữ liệu. Các ứng dụng này chỉ đơn giản là xuất dữ liệu trong khối.

• Giấy phép - Node.js được phát hành theo giấy phép MIT

Một số bộ phận quan trọng của NodeJS



Các trường hợp để sử dụng NodeJS

• I/O bound Applications

• Data Streaming Applications

• Data Intensive Real time Applications (DIRT)

• JSON APIs based Applications

• Single Page Applications

Ưu điểm của NodeJS

• JSON APIs : Bởi lẽ REST/JSON APIs gọn nhẹ là điều khiến NodeJS tỏa sáng. Với cơ chế event-driven, non-blocking I/O(Input/Output) và mô hình kết hợp với JavaScript là sự lựa chọn tuyệt vời cho các dịch vụ web làm bằng JSON.

• Ứng dụng trên 1 trang : Nếu bạn định viết 1 ứng dụng thể hiện trên 1 trang (Gmail?) NodeJS rất phù hợp để làm. Với khả năng xử lý nhiều request đồng thời thời gian phản hồi nhanh. Các ứng dụng bạn định viết không muốn nó tải lại trang, gồm rất nhiều request từ người dùng cần sự hoạt động nhanh để thể hiện sự chuyên nghiệp thì NodeJS sẽ là sự lựa chọn của bạn.

• Shelling tools unix: NodeJS sẽ tận dụng tối đa Unix để hoạt động. Tức là NodeJS có thể xử lý hàng nghìn Process và trả ra 1 luồng khiến cho hiệu suất hoạt động đạt mức tối đa nhất và tuyệt vời nhất.

• Streaming Data (Luồng dữ liệu): Các web thông thường gửi HTTP request và nhận phản hồi lại (Luồng dữ liệu). Giả sử sẽ cần xử lý 1 luồng dữ liệu cực lớn, NodeJS sẽ xây dựng các Proxy phân vùng các luồng dữ liệu để đảm bảo tối đa hoạt động cho các luồng dữ liệu khác.

• Ứng dụng Web thực : Giả sử bạn xây dựng 1 ứng dụng chat, feed ... Facebook, Twitter là điển hình cho Web thực. NodeJS làm khá tốt điều đó.

Một số nhược điểm của NodeJS

• Ứng dụng nặng tốn tài nguyên: Nếu bạn cần xử lý các ứng dụng tốn tài nguyên CPU như encoding video, convert file, decoding encryption... hoặc các ứng dụng tương tự như vậy thì không nên dùng NodeJS (lý do: NodeJS được viết bằng C++ & Javascript, nên phải thông qua thêm 1 trình biên dịch của NodeJS sẽ lâu hơn 1 chút). Trường hợp này bạn hãy viết một Addon C++ để tích hợp với NodeJS để tăng hiệu suất tối đa(việc tích hợp rất thân thiện và nhanh chóng).

• NodeJS và ngôn ngữ khác: NodeJS, PHP, Ruby, Python .NET ...thì việc cuối cùng là phát triển các App Web. NodeJS mới sơ khai như các ngôn ngữ lập trình khác. Vậy nên bạn đừng hi vọng NodeJS sẽ hơn PHP,Ruby,Python... ở thời điểm này. Nhưng với NodeJS bạn có thể có 1 ứng dụng như mong đợi, điều đó là chắc chắn (perfect).

• Với những gì các ngôn ngữ trước đang có(cộng đồng lâu năm, Framework, CMS, Opensource...) Nếu bạn/doanh nghiệp chưa biết về NodeJS thì việc cần xây dựng dự án quan

trọng, kinh doanh phát triển trên NodeJS sẽ không phải lựa chọn bây giờ. • NoSQL + Nodejs + Noob: Với NodeJS, NoSQL thì là sự kết hợp hoàn hảo.

• Bạn là người có kinh nghiệm với các ngôn ngữ lập trình để phát triển các dự án. Bạn biết được NodeJS qua tin tức, báo chí, bạn bè... Bạn quyết định xây dựng dự án bằng NodeJS. Nhưng khi gặp sự cố rủi ro xây dựng dự án với NodeJS đồng thời quay lưng luôn. Hãy đừng đổ lỗi cho công nghệ bạn đang dùng mà hãy hiểu rằng "Bạn chưa hiểu được NodeJS !".

Muốn nghiên cứu chuyên sâu hơn nữa có thể tham khảo link sau :

NodeJS #Sơ lược về Socket và Socket.IO Trước tiên, Socket là một công nghệ. Đừng nhầm lẫn giữa Socket.IO và Socket. Socket.IO không phải là mô hình Socket duy nhất hiện nay và cũng không phải là mô hình Web-Socket duy nhất hiện nay.Socket là cách bạn tổ chức mô hình Client-Server để một trong 2 bên luôn trong tình trạng sẵn sàng trả lời bên kia và ngược lại. Để đảm bảo việc này, kết nối giữa Client và Server phải ở trạng thái “keep-alive” và phải luôn xảy ra quá trình đồng bộ giữa Client-Server. Socket sẽ mang lại khả năng trả lời tức thì từ một trong 2 bên khi bên kia đưa ra một sự kiện, thay vì phải thực thi lại một loạt các thủ tục kết nối phức tạp như trước, và ứng dụng của bạn sẽ trở thành ứng dụng thời gian thực ví dụ: Yahoo Messenger, Skype v.v… đều là các ứng dụng được xây dựng theo mô hình Socket.

Trong lập trình web trước đây, việc xây dựng Client-Server theo mô hình Socket phải thông qua các phần mềm thứ . Vì mô hình Socket không phù hợp với các ngôn ngữ lập trình Server như: PHP, ASP.NET, JSP v.v… Các ngôn ngữ này luôn làm việc theo cách: Die ngay connection khi Server trả lời Client xong. Tuy nhiên, mình nhấn mạnh: Chúng ta có thể làm được Web-Socket với bất kỳ ngôn ngữ lập trình nào. Chỉ có điều, với các ngôn ngữ cũ, việc làm này cần bạn phải am hiểu các giao thức http, tcp; hiểu thế nào là 1 request header, v.v… Node.js là một ngôn ngữ mới, xây dựng thuần túy bằng JavaScript. Đây là một điểm lợi thế của Node.js để lập trình Web-Socket:

• Thứ nhất: JavaScript là ngôn ngữ lập trình hướng sự kiện, mà trong lập trình thời gian thực, cách tiếp cận bằng lập trình sự kiện là cách tiếp cận khôn ngoan nhất.

• Thứ hai: Node.js chạy non-blocking việc hệ thống không phải tạm ngừng để xử lý xong một request sẽ giúp cho Server trả lời Client gần như ngay tức thì.

• Thứ ba: lập trình Socket yêu cầu bạn phải xây dựng được mô hình lắng nghe - trả lời từ cả 2 bên. Nói khác đi, vai trò của Client và Server phải tương đương nhau, mà client thì chạy bằng JavaScript, nên nếu server cũng chạy bằng javascript nữa, thì việc lập trình sẽ dễ dàng và thân thiện hơn.

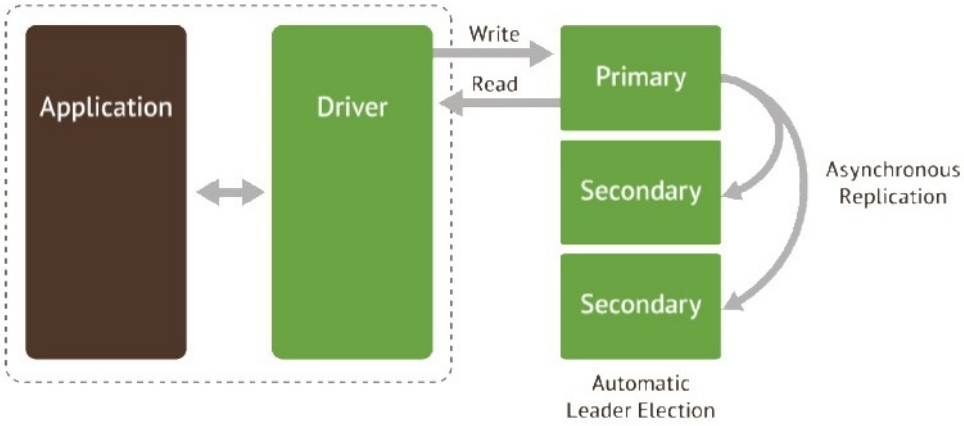
• Chính vì những đặc điểm này, Socket.IO ra đời. Tuy nhiên, khi bạn thực sự am hiểu về Node.js, http request header, bạn hoàn toàn có thể viết một socket cho riêng mình. #Đôi nét về MongoDB ##MongoDB là gì ? Hiểu một cách nôm na thì MongoDB là một mã nguồn mở và là một tập tài liệu dùng cơ chế NoSQL để truy vấn, nó được viết bởi ngôn ngữ C++. Chính vì được viết bởi C++ nên nó có khả năng tính toán với tốc độ cao chứ không giống như các hệ quản trị CSDL hiện nay.

Nếu như bạn biết sử dụng JSON thì trong MongoDB cũng có cấu trúc lưu trữ tương tự, chính vì thế nó có hiệu suất cao, tương tác nhanh và khả năng mở rộng tốt, nó hoạt động trên khái niệm collection và document.

• Collection trong MongoDB là nhóm các tài liệu (document), nó tương đương với một bảng (table) trong CSDL thông thường nên mỗi collection sẽ thuộc về một database duy nhất. Tuy nhiên nó có một sự khác biệt đó là nó không có ràng buộc Relationship như các hệ quản trị CSDL khác nên việc truy xuất rất nhanh, chính vì thế mỗi collection có thể chứa nhiều thể loại khác nhau không giống như table trong hệ quản trị mysql là các field cố định.

• Document trong MongoDB có cấu trúc tương tự như kiểu dữ liệu JSON, nghĩa là sẽ có các cặp (key => giá trị) nên nó có tính năng động rất lớn. Document ta có thể hiểu nó giống như các record dữ liệu trong MYSQL, tuy nhiên nó có sự khác biệt là các cặp (key => value) có thể không

giống nhau ở mỗi document. ##MongoDB hoạt động như thế nào



MongoDB hoạt động dưới một tiến trình ngầm service luôn mở một cổng (Cổng mặc định là 27017) để lắng nghe các yêu cầu truy vấn, thao tác từ các ứng dụng gửi vào sau đó mới tiến hành xử lý. Mỗi một bản ghi của MongoDB được tự động gắn thêm một field có tên "_id" thuộc kiểu dữ liệu ObjectId mà nó quy định để xác định được tính duy nhất của bản ghi này so với bản ghi khác, cũng như phục vụ các thao tác tìm kiếm và truy vấn thông tin về sau. Trường dữ liệu "_id" luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất. Mỗi khi có một truy vấn dữ liệu, bản ghi được cache (ghi đệm) lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng. Khi có yêu cầu thêm/sửa/xóa bản ghi, để đảm bảo hiệu suất của ứng dụng mặc định MongoDB sẽ chưa cập nhật xuống ổ cứng ngay, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng. ##Ưu điểm của MongoDB : • Dữ liệu lưu trữ phi cấu trúc, không có tính ràng buộc, toàn vẹn nên tính sẵn sàng cao, hiệu suất lớn và dễ dàng mở rộng lưu trữ. • Dữ liệu được caching (ghi đệm) lên RAM, hạn chế truy cập vào ổ cứng nên tốc độ đọc và ghi cao ##Nhược điểm của MongoDB : • Không ràng buộc, toàn vẹn nên không ứng dụng được cho các mô hình giao dịch yêu cầu độ chính xác cao.

• Không có cơ chế transaction (giao dịch) để phục vụ các ứng dụng ngân hàng.

• Dữ liệu được caching, lấy RAM làm trọng tâm hoạt động vì vậy khi hoạt động yêu cầu một bộ nhớ RAM lớn.

• Mọi thay đổi về dữ liệu mặc định đều chưa được ghi xuống ổ cứng ngay lập tức vì vậy khả năng bị mất dữ liệu từ nguyên nhân mất điện đột xuất là rất cao.