¿Simplemente qué es Node.js?

Un servidor listo para codificar

Michael Abernethy 14-06-2011

Node es un intérprete Javascript del lado del servidor que cambia la noción de cómo debería trabajar un servidor. Su meta es permitir a un programador construir aplicaciones altamente escalables y escribir código que maneje decenas de miles de conexiones simultáneas en una sólo una máquina física.

Introducción

Si usted ha oído acerca de Node, o si ha leído algún artículo destacando lo increíble que es, tal vez se esté preguntando "¿Simplemente qué es Node.js?". Tal vez usted *todavía* tenga preguntas sobre lo que es Node, después de leer su propia página principal. Node definitivamente no es para todos los programadores, pero podría ser la respuesta adecuada para algunos.

Este artículo tratará de responder a los que es Node.js, presentando un breve contexto de los problemas que resuelve, cómo funciona, cómo ejecutar una aplicación simple y, finalmente, dónde Node es una buena solución. No tratará cómo escribir una aplicación Node complicada ni será un tutorial minucioso sobre Node. Leer este artículo le ayudará a decidir si debe continuar aprendiendo sobre Node para utilizarlo en su propio negocio.

¿Qué problema resuelve Node?

La meta número uno declarada de Node es "proporcionar una manera fácil para construir programas de red escalables". ¿Cuál es el problema con los programas de servidor actuales? Hagamos cuentas. En lenguajes como Java™ y PHP, cada conexión genera un nuevo hilo que potencialmente viene acompañado de 2 MB de memoria. En un sistema que tiene 8 GB de RAM, esto da un número máximo teórico de conexiones concurrentes de cerca de 4.000 usuarios. A medida que crece su base de clientes, si usted desea que su aplicación soporte más usuarios, necesitará agregar más y más servidores. Desde luego, esto suma en cuanto a los costos de servidor del negocio, a los costos de tráfico, los costos laborales, y más. Además de estos costos están los costos por los problemas técnicos potenciales — un usuario puede estar usando diferentes servidores para cada solicitud, así que cualquier recurso compartido debe almacenarse

en todos los servidores. Por todas estas razones, el cuello de botella en toda la arquitectura de aplicación Web (incluyendo el rendimiento del tráfico, la velocidad de procesador y la velocidad de memoria) era el número máximo de conexiones concurrentes que podía manejar un servidor.

Node resuelve este problema cambiando la forma en que se realiza una conexión con el servidor. En lugar de generar un nuevo hilo de OS para cada conexión (y de asignarle la memoria acompañante), cada conexión dispara una ejecución de evento dentro del proceso del motor de Node. Node también afirma que nunca se quedará en punto muerto, porque no se permiten bloqueos y porque no se bloquea directamente para llamados E/S. Node afirma que un servidor que lo ejecute puede soportar decenas de miles de conexiones concurrentes.

Entonces, ahora que usted tiene un programa que puede manejar cientos de miles de conexiones concurrentes, ¿qué puede usted construir en realidad con Node? Sería extraordinario si usted tuviera una aplicación Web que necesitara de toda esta cantidad de conexiones. Ese es uno de esos problemas del tipo "si usted tiene este problema, no es un problema". Antes de pasar a ello, observemos cómo funciona Node y cómo está diseñado que se ejecute.

Lo que Node definitivamente no es

Sí, Node es un programa de servidor. Sin embargo, el producto base de Node definitivamente *No* es como Apache o Tomcat. Esos servidores básicamente son productos para servidor listos para instalar y que están listos para implementar aplicaciones instantáneamente. Usted podría tener un servidor estar listo y en operación en un minuto con estos productos. Node definitivamente no es esto. De forma similar a como Apache puede agregar un módulo PHP para permitir a los desarrolladores crear páginas Web dinámicas, y un módulo SSL para conexiones seguras, Node también tiene el concepto de módulos que se pueden agregar a su núcleo mismo. Literalmente hay cientos de módulos de los que se puede escoger con Node, y la comunidad es bastante activa en cuanto a producir, publicar y actualizar docenas de módulos por día. Hablaremos sobre toda la parte de módulos de Node más adelante en este artículo.

Cómo funciona Node

Node ejecuta V8 JavaScript. Espere... ¿qué? ¿JavaScript en el servidor? Sí, leyó correctamente. El JavaScript del lado del servidor puede ser un concepto nuevo para cualquiera que haya trabajado exclusivamente con JavaScript del lado del cliente, pero la idea en sí no es tan inverosímil — ¿por qué no utilizar el mismo lenguaje de programación que usted usa en el cliente del lado del servidor?

¿Qué es el V8? El motor V8 JavaScript es el motor JavaScript subyacente que Google usa con su navegador Chrome. Pocas personas piensan en lo que en realidad sucede con JavaScript en el cliente. Bien, un motor JavaScript en realidad interpreta el código y lo ejecuta. Con el V8, Google creó un intérprete ultra-rápido escrito en C++, con otro aspecto único: usted puede descargar el motor e incorporarlo a *cualquier* aplicación que desee. No está restringido a ejecutarse en un navegador. Así, Node en realidad usa el motor V8 JavaScript escrito por Google y le da otro propósito para usarlo en el servidor. ¡Perfecto! Para qué crear un nuevo lenguaje cuando ya hay una buena solución disponible.

Programación orientada por eventos

A muchos programadores se les ha hecho creer que la programación orientada a objetos es el diseño perfecto de programación y que no deben usar nada más. Node utiliza lo que se conoce como modelo de programación orientado por eventos.

Listado 1. Programación orientada por eventos del lado del cliente con jQuery

```
// jQuery code on the client-side showing how Event-Driven programming works

// When a button is pressed, an Event occurs - deal with it
// directly right here in an anonymous function, where all the
// necessary variables are present and can be referenced directly
$("#myButton").click(function(){
     if ($("#myTextField").val() != $(this).val())
         alert("Field must match button text");
});
```

El lado del servidor realmente no es tan diferente del lado del cliente. Es verdad, no se están presionando botones, y no se está ingresando texto en campos, pero a un nivel superior, *están* sucediendo eventos. Se realiza una conexión — ¡evento! Se reciben datos a través de la conexión — ¡evento! Se dejan de recibir datos por la conexión — ¡evento!

¿Por qué este tipo de configuración es ideal para Node? JavaScript es un gran lenguaje para programación orientada por eventos, porque permite funciones y cierres anónimos, y más importante, la sintaxis es similar para casi cualquier persona que haya codificado. las funciones de devolución de llamado que se llaman cuando ocurre un evento pueden escribirse en el mismo punto en el que usted captura el evento. Fácil de codificar, fácil de mantener. No hay infraestructuras complicadas Orientadas a Objeto, no hay interfaces, no hay potencial para sobre-arquitectura de nada. Simplemente esperar por un evento, escribir una función de devolución de llamado, ¡y se ha resuelto todo!

Aplicación Node de ejemplo

¡Finalmente vamos a ver algo de código! Reunamos todas las cosas sobre las que hemos tratado y creemos nuestra primera aplicación Node. Como hemos visto que Node es ideal para el manejo de aplicaciones de alto tráfico, creemos una aplicación Web bastante simple, construida para máxima velocidad. Estas son las especificaciones para nuestra aplicación de muestra pasadas por el "jefe": Cree una API RESTful generadora de números aleatorios. La aplicación debe recibir una entrada, un parámetro llamado "number". La aplicación retornará un número aleatorio que esté entre 0 y este parámetro, y retornará ese número generado a quien hizo el llamado. Y, como el "jefe" espera que esta sea una aplicación masivamente popular, deberá manejar 50.000 usuarios concurrentes. Observemos el código:

Listado 2. Generador Node de números aleatorios

```
// these modules need to be imported in order to use them.
// Node has several modules.  They are like any #include
// or import statement in other languages
var http = require("http");
var url = require("url");

// The most important line in any Node file.  This function
```

```
// does the actual process of creating the server.  Technically,
// Node tells the underlying operating system that whenever a
// connection is made, this particular callback function should be
// executed.  Since we're creating a web service with REST API,
// we want an HTTP server, which requires the http variable
// we created in the lines above.
// Finally, you can see that the callback method receives a 'request'
// and 'response' object automatically.  This should be familiar
// to any PHP or Java programmer.
http.createServer(function(request, response) {

     // The response needs to handle all the headers, and the return codes
     // These types of things are handled automatically in server programs
     // like Apache and Tomcat, but Node requires everything to be done yourself
     response.writeHead(200, {"Content-Type": "text/plain"});

     // Here is some unique-looking code.  This is how Node retrives
     // parameters passed in from client requests.  The url module
     // handles all these functions.  The parse function
     // deconstructs the URL, and places the query key-values in the
     // query object.  We can find the value for the "number" key
     // by referencing it directly - the beauty of JavaScript.
     var params = url.parse(request.url, true).query;
     var input = params.number;

     // These are the generic JavaScript methods that will create
     // our random number that gets passed back to the caller
     var numInput = new Number(input);
     var numOutput = new Number(Math.random() * numInput).toFixed(0);

     // Write the random number to response
     response.write(numOutput);

     // Node requires us to explicitly end this connection.  This is because
     // Node allows you to keep a connection open and pass data back and forth,
     // though that advanced topic isn't discussed in this article.
     response.end();

   // When we create the server, we have to explicitly connect the HTTP server to
   // a port.  Standard HTTP port is 80, so we'll connect it to that one.
}).listen(80);

// Output a String to the console once the server starts up, letting us know everything
// starts up correctly
console.log("Random Number Generator Running...");
```

Iniciando esta aplicación

Ponga el código anterior en un archivo llamado "random.js". Ahora, para comenzar esta aplicación y ejecutarla (creando así el servidor HTTP y escuchando las conexiones en el puerto 80), simplemente corra el siguiente comando en su prompt de comandos: `% node random.js`. Así es como se verá cuando usted sepa que el servidor estará listo y en funcionamiento.

```
root@ubuntu:/home/moila/ws/mike# node random.js
Random Number Generator Running...
```

Accediendo a la aplicación

La aplicación está lista y funcionando. Node está escuchando a cualquier conexión en este momento, así que vamos a probar la aplicación. Como hemos creado una API RESTful simple, podemos accesar a la aplicación usando nuestro navegador Web. Digite la siguiente dirección (asegúrese de haber completado el paso anterior), http://localhost/?number=27.

La ventana de su navegador cambiará a un número aleatorio entre 0 y 27. Presione recargar en su navegador y obtendrá otro número aleatorio. Listo, ¡esta es su primera aplicación Node!

Node, ¿para qué sirve?

Entonces, después de leer todo sobre Node, usted tal vez pueda responder la pregunta "Simplemente qué es Node?" pero usted puede quedar con la duda sobre "¿En qué puedo utilizar Node?" Esa es una pregunta importante ya que hay algunas cosas para las que Node es realmente bueno.

Para qué sirve

Como ha visto hasta ahora, Node está extremadamente bien diseñado para situaciones en que usted esté esperando una gran cantidad de tráfico y donde la lógica del lado del servidor y el procesamiento requeridos, no sean necesariamente grandes antes de responder al cliente. Aquí hay algunos buenos ejemplos en donde Node haría un gran trabajo:

- **Una API RESTful**
  Un servicio Web que proporcione una API RESTful toma algunos parámetros, los interpreta, arma una respuesta y descarga esa respuesta (usualmente una cantidad relativamente pequeña de texto) de vuelta al usuario. Esta es una situación ideal para Node, porque puede construirse para que maneje decenas de miles de conexiones. Tampoco requiere una gran cantidad de lógica y básicamente sólo busca valores de una base de datos y los reúne como una respuesta. Como la respuesta es una pequeña cantidad de texto y la solicitud entrante es una pequeña cantidad de texto, el volumen de tráfico no es alto, y una máquina probablemente puede manejar las demandas de API de incluso la API de la más ocupada de las empresas.
- **Fila de Twitter**
  Piense en una compañía como Twitter que recibe tweets y los escribe en una base de datos. Literalmente hay miles de tweets llegando cada segundo y la base de datos posiblemente no puede seguir el ritmo del número de escrituras necesarias durante los horarios pico de uso. Node se convierte en una pieza clave de la solución a este problema. Como hemos visto, Node puede manejar decenas de miles de tweets entrantes. Luego puede escribirlos rápida/fácilmente en un mecanismo de cola en memoria (memcached, por ejemplo), desde donde otro proceso separado puede escribirlos en la base de datos. El rol de Node en esto es reunir rápidamente el tweet y pasar esta información hacia otro proceso responsable de escribirlo. Imagine otro diseño — un servidor PHP normal que intente manejar escrituras en la base de datos misma — cada tweet podría causar una pequeña demora mientras se escribe en la base de datos, dado que el llamado de base de datos estaría bloqueando. Una máquina con este diseño sólo podría manejar 2.000 tweets entrantes por segundo, debido a la latencia de base de datos. A un millón de tweets por segundo, usted estaría hablando de 500 servidores. Node, en cambio, maneja cada conexión y no causa bloqueo, permitiéndole capturar tantos tweets como se le puedan arrojar. Una máquina nodo capaz de manejar 50.000 tweets por segundo, y usted estaría hablando de sólo 20 servidores.
- **Estadísticas de videojuegos**
  Si usted alguna vez jugó un juego como Call of Duty on-line, algunas cosas le habrán llamado la atención inmediatamente cuando observó las estadísticas del juego,

principalmente el hecho de que deben estar rastreando toneladas de información sobre el juego para poder producir tal nivel de estadísticas. Luego, multiplique esto por los millones de personas que lo juegan en cualquier momento, y tendrá una idea de la inmensa cantidad de información que se genera con bastante rapidez. Node es una buena solución para este escenario, porque puede capturar los datos que están generando los juegos, hacer un mínimo de consolidación con ellos y luego ponerlos en una fila para escribirlos en una base de datos. Parecería algo tonto dedicar todo un servidor a rastrear cuántas balas disparan las personas en los juegos, lo cual podría ser el límite útil si usted utilizara un servidor como Apache, pero parecería menos tonto si en lugar de ello usted pudiera dedicar un solo servidor a rastrear casi todas las estadísticas de un juego, como usted puede llegar a hacerlo con un servidor que ejecute Node.

Módulos Node

Aunque originalmente no era un tema planeado en el artículo, debido a la demanda popular, el artículo se ha expandido para incluir una breve introducción a los Node Modules y al Node Package Manager. Así como a lo que las personas se han acostumbrado al trabajar con Apache, usted puede expandir la funcionalidad de Node instalando módulos. No obstante, los módulos que usted puede utilizar con Node mejoran *en gran medida* el producto, tanto, que es poco probable que haya alguien que utilice Node sin instalar por lo menos algunos módulos. Así de importantes se han tornado los módulos, hasta el punto de convertirse en parte esencial del producto completo.

En el aparte de Recursos, incluyo un enlace a la página de módulos, donde están listados y disponibles todos los módulos posibles. Como una muestra rápida de las posibilidades, estos incluyen un módulo para escribir páginas creadas dinámicamente (como PHP), un módulo para facilitar el trabajo con MySQL, un módulo para ayudar con WebSockets, y un módulo para asistir en el análisis de texto y de parámetros, entre docenas de módulos disponibles. No entraré en detalles sobre los módulos, porque como mencioné, este sólo es un artículo de visión general que le ayuda a entender si Node es algo que usted debería buscar más adelante, pero es posible, si decide utilizarlo más adelante, que definitivamente también vaya a trabajar con los módulos disponibles.

Adicionalmente, Node presenta el Node Package Module, que es una forma integrada de instalar y administrar los módulos Node que esté usando. Este maneja automáticamente dependencias, de manera que usted puede estar seguro(a) de que cualquier módulo que usted desee instalar se instalará correctamente con todas sus partes necesarias. También sirve como una forma para publicar sus propios módulos en la comunidad Node, si usted opta por vincularse y escribir su propio módulo. Piense en el NPM como una forma fácil para expandir la funcionalidad de Node sin tener que preocuparse por desconfigurar su instalación Node. De nuevo, si usted opta por continuar avanzando en Node, el NPM será parte vital de su solución Node.

Conclusión

Nota del Editor

La versión inicialmente publicada de este artículo generó bastantes comentarios por parte de la comunidad, sobre los diferentes puntos que se presentaron. Desde ese entonces el autor

ha revisado este artículo con esas ideas en mente. Este tipo de revisión y discusión por lo pares es parte vital del mundo de fuente abierta. Gracias a aquellos(as) que aportaron sus comentarios constructivos.

Como todos los proyectos de fuente abierta, Node.js continuará evolucionando y los desarrolladores descubrirán nuevos recursos y técnicas para superar cualquier número de limitaciones. Como siempre, alentamos a nuestros usuarios a que prueben la tecnología por sí mismos.

La pregunta que muchos de ustedes tenían al comienzo de este artículo "¿Simplemente qué es Node.js?" debería quedar respondida después de leerlo. Usted debe poder explicar en unas pocas frases sencillas y concisas lo que es Node.js. Si usted puede hacerlo, entonces estará un paso adelante de casi cualquier otro programador. Muchas personas con las que he hablado sobre Node han estado confundidas con respecto a lo que hace exactamente. Ellas están, y es comprensible, pensando en modo Apache — (— un servidor es una aplicación en la que usted descarga sus archivos HTML y todo funciona). Como la mayoría de los programadores están acostumbrados a Apache y a lo que hace, la forma más fácil de describir Node es comparándolo con Apache. Node es un programa que puede hacer todo lo que hace Apache (con algunos módulos), pero que también puede hacer mucho más, al ser una plataforma JavaScript extensible desde la cual usted puede construir.

En este artículo usted ha visto cómo Node cumple con sus metas de proporcionar servidores altamente escalables. Utiliza un motor JavaScript extremadamente rápido de Google, el motor V8. Utiliza un diseño Orientado por Eventos para mantener el código al mínimo y fácil de leer. Todos estos factores conducen a la meta deseada por Node — es relativamente fácil escribir una solución masivamente escalable.

Tan importante como entender lo que Node *es*, también es importante entender lo que Node *no es*. Node no es simplemente un reemplazo de Apache que instantáneamente vaya a hacer más escalable su aplicación Web PHP. Eso no podría estar más lejos de la verdad. Todavía es muy temprano en la vida de Node, pero está creciendo extremadamente rápido, la comunidad está involucrada muy activamente, se está creando una gran cantidad de módulos, y este producto en crecimiento podría estar en su negocio dentro de un año.

Temas relacionados

- El argumento Página principal de Node.js es el punto de lanzamiento para aprender sobre la aplicación.
- Descargue Node.js aquí. También necesitará Python .
- Navegue la página de la API Node.js. Note que la sintaxis puede cambiar de release a release, así que verifique dos veces la versión que haya descargado y la APi en la que esté navegando.
- Consulte la página de módulos Node que lista todos los módulos que se pueden usar en Node.
- Busque el NPM para facilitar la expansión de la funcionalidad de su instalación Node.
- Vea la lista en constante evolución de open source software packages en Wikipedia.
- Siga a IBM Developer en Twitter.