**What I Wish I Had Known Before Scaling Uber to 1000 Services**

El video comienza mostrando al expositor, Matt Ranney, explicando que el tema del cual se estará hablando durante la exposición son las cosas que el deseo haber sabido antes del gran escalamiento/crecimiento que ha tenido Uber. Lo primero que se hace es mostrar un par de videos mostrando el patrón de explosión de crecimiento que han tenido dos ciudades de China, y lo compara con el hecho de que Uber cuando el ingreso hace un año contaba con tan solo 200 ingenieros, teniendo en la actualidad cerca de 2000, mostrándose un incremento de x10.

La conferencia continua con Matt exponiendo sobre los errores que se han cometido a lo largo del camino hacia alcanzar su estado actual, y que todo lo que él ha aprendido, se lo daría de consejo a su “yo” más joven de hace un año y medio, pero que este seguramente no lo escucharía.

A lo largo de la exposición se da a entender que Uber se encuentra dentro de una “batalla” por todo el mundo, dentro de un modelo de negocio donde el ganador se lo lleva todo, y es en este contexto donde el crecimiento debe ser lo más rápido posible. Para este tipo de cuestiones Matt menciona que los micro servicios han sido la solución perfecta para lo que Uber está buscando lograr.

**Micro servicios. -** En su exposición, Matt Ranney que Uber opta por utilizar una gran cantidad de micro servicios en lugar de tener lo que denomina como “monolitos” en el sistema. Además, menciona otras cosas buenas y los costos que se les tiene asignados.

* Lo bueno. – Los micro servicios permiten que los equipos se formen rápidamente y se ejecuten de forma independiente. Se es dueño de su propio tiempo de actividad, uno mismo ejecuta lo que escribe. Se puede usar la que uno piense es la “mejor” herramienta para esa actividad determinada
* Los costos evidentes. - Los microservicios se ejecutan dentro de lo que es un sistema distribuido, lo cual es más difícil que trabajar con un monolito. Todo es un RPC. ¿Qué pasa si se rompe? ¿Cómo se solucionan los problemas? ¿Cómo averigua en qué parte de la cadena de servicios se produjo la ruptura?
* Los costos menos evidentes. – Todo es una compensación, a cambio de todos estos micro servicios obtienes algo, pero se cede algo también. Uno puede optar por crear un nuevo servicio en lugar de arreglar algo que ya está roto, lo cual en algún momento puede comenzar a ser un factor en el costo.

**Lenguajes**

En un principio Uber fue completamente tercerizado. Actualmente, con el uso de microservicios, los equipos pueden escribir en diferentes idiomas y aún pueden comunicarse entre ellos. Sin embargo, esto tiene algunos costos, como lo pueden ser: difícil de compartir el código, difícil de mover entre los equipos y además el tener distintos idiomas puede fragmentar la cultura.

**El costo de RPC**

Los equipos se comunican entre sí utilizando RPC. A gran escala, con muchas y muchas personas uniéndose rápidamente, las debilidades de HTTP comienzan a aparecer. Los servidores no son navegadores. Cuando se habla a través de un centro de datos, tiene mucho más sentido tratar todo como una llamada a función en lugar de una solicitud web.

**Repositorios**

La tendencia actualmente es tener muchos repositorios para tener muchos módulos pequeños, los cuales son fáciles de intercambiar. Sin embargo, tener un solo repositorio también puede ser excelente porque permite realizar cambios transversales, si se deseara realizar un cambio, es fácil acceder y navegar por todo el código que debe modificarse. Uber cuenta con más de 8000 repositorios en git.

**Desempeño**

El rendimiento definitivamente va a surgir dado cuán dependientes están los microservicios entre sí. Los RPC son caros, y especialmente cuando hay varios idiomas, la respuesta sobre cómo entiende su rendimiento depende totalmente de las herramientas de lenguaje y las herramientas son todas diferentes.

**Fanout**

Fanout causa muchos problemas de rendimiento. Imagine un servicio típico que responde el 99% del tiempo en 1 ms. 1% de las veces responde en un segundo. Todavía no es tan malo. 1% de las veces los usuarios obtendrán la lentitud. Ahora digamos que los servicios comienzan a tener un gran fanout, llamando a muchos otros servicios. La probabilidad de experimentar tiempos de respuesta lentos aumenta rápidamente. Use 100 servicios y 63% del tiempo de respuesta en al menos 1 segundo (1.0 - .99 ^ 100 = 63.4%).

**Logging**

Múltiples idiomas dificultan el registro. Con un grupo de diferentes idiomas, muchos equipos y mucha gente nueva, la mitad del equipo de ingeniería ha estado presente por menos de 6 meses, todos podrían tender a conectarse de maneras muy diferentes. Se deben proporcionar herramientas que sean tan obvias y fáciles de usar que las personas no tendrían otra manera para obtener un registro.

**Prueba de carga**

Es un problema generar datos de prueba realistas para probar todas las partes del sistema. Por lo cual estas se deben realizar fuera de los horarios de mayor concurrencia. Lo que realmente se requiere hacer es cargar todos los servicios a través de todo el tiempo porque muchos errores solo aparecen cuando el tráfico alcanza su punto máximo.

**Pruebas de fallo**

No a todo el mundo le gustan este tipo de pruebas, sin embargo, estas pruebas deben darse así a uno le guste o no ya que es parte común de la producción

**Migraciones**

Las cosas viejas todavía tienen que funcionar cuando se realiza una migración. Los mandatos para migrar son malos, ya que nadie quiere que le digan que tienen que adoptar un nuevo sistema. Se debe ofrecer un nuevo sistema mucho mejor a alguien en lugar de obligarlo a cambiar solo porque la organización requiere del cambio.

**Política**. – Se debe respetar la propiedad Empresa > Equipo > Uno mismo