G: VStar corre en una sola máquina, he leído de lo memory mapping y todo esto, lo que no me queda muy claro es dónde están almacenados los datos, me había dicho Eduardo que están almacenados en un SAN?
E: No, no necesariamente. NO es indispensable en SAN. El comentario de Eduardo seguramente fue en la dirección que para la distribución, cuando quieres manejar un ambiente distribuido para VStar la solución idónea es utilizando un SAN. Utilizando un sistema de operación distribuido como el de IBM y eso es por ejemplo lo que nuestro cliente principal utiliza en Austria, un SAN donde cada una de las máquinas están conectadas a todos los discos en términos generales y no les importa a los procesos en donde están corriendo. Mira, lo interesante de VStar es ésto, voy a tartar de hacer un resumen. Primero, los datos los guardamos como cualquier otra base de datos, los guardamos en nuestras estructuras, en el disco. Lo diferente de nosotros es que nuestras estructuras no son nada propietarias. VST res un agregado de una extension de los lenguajes tipo APL. Recuerdas APL?
G: Nunca lo usé
E: Bueno, APL mientras hubo main frames y grandes cantidades de memoria fue una solución muy importante, sobre todo en ambientes financieros. Cuando salieron las micros, a finales de los 90s y la memoria se volvió pequeña, ambientes como APL perdieron. A lo largo de los 90s y comienzo del 2000 salieron los dos descendientes, uno se llama J y el otro se llama K. La gente de K creó su propia base de datos que se llama KX y es una base de datos estructuralmente muy similar a VStar, solamente que está basada en K, mientras que VStar está basada en J. La otra diferencia es que J es una compañía independiente de nosotros pero estamos muy relacionados, de hecho el jefe de ingeniería de J software es práctimente un arquitecto de VStar. En cambio, K ellos se dedican a hacer el lenguaje y la base de datos juntos. Pero el mercado en el que K se ha concentrado, ellos lo dominan, prácticamente ellos son dueños de lo que se llama High Speed y Event Processing para billones y billones de datos, por ej. todo lo que son la NASDAQ level 2. Ellos tienen una gran parte de ese mercado, en NY y en Zurich, etc. La diferenciación con nosotros es que VSTar, basada en J, J en sí es un lenguaje mucho más versatil que K. K está hecho específicamente para series de tiempo y no tiene práctimente extensiones de ningún tipo, por ejemplo, no tiene numeros complejos, no tiene

operaciones matriciales o geométricas para calcular senos, cosenos , etc. Y J es un lenguaje totalmente matemático, con todas las operaciones de alta precision, además de muy optimizadas, lo que hace a la base de datos de un uso mucho más general. Bueno, ahora nosotros ¿qué fue lo que hicimos al crear VStar? Nos aprovechamos del hecho que estamos arriba de un lenguaje totalmente basado en arreglos, me refiero a J, de tal manera de que no tuvimos que crear nuestras propias librerías para el manejo de operaciones, por ejemplo multidimensionales, operaciones para el reverso de una matriz, operaciones simplemente para buscar un elemento en un cubo, a diferencia de Vértika, que ellos tuvieron que hacerlo porque están escritos en C. Ahora J está escrito en C, y tu puedes incorporar código native en J y en VStar en cualquier momento, de hecho la interfase es prácticamente transparente. La diferencia primordial es que nuestros datos están almacenados en estructuras simples, por ejemplo una columna de cada tabla tuya es un simple archivo que tiene una secuencia de valores binarios. ¿Qué tipo de valor? Depende del tipo de datos que estés guardando. Somos exclusivamente 64 bit, puedes correrla en una máquina de 32 bit, pero es práctimente un juguete, es para propósitos de demo en una Lap. La version que usamos para trabajo es la de 64. De la misma manera, corremos bien en Windows y hay operaciones que funcionan bastante bien ahí, pero en terminos generales el performance, sobre todo donde involucras discos, el performance en Linux es dos o tres veces mayor. Pero insisto, hay ciertas cosas que funcionan en Windows y no están incluidas en Linux, simplemente por la generalidad de las cosas. Ahora, cada columna es un archivo, eso tuvo implicaciones muy importantes, porque cada tabla es un directorio, entonces al hacer las cosas de esta manera, resulta que la seguridad con VStar, que a lo major para ustedes la seguridad no es tan importante como para un Banco, pero para darte una idea de porqué un Banco podría comprarle a una compañía muy chica, es que nosotros no nos dedicamos a implementar la seguridad, aprovechamos la seguridad del sistema operative subyacente. Entonces tu asignas la seguridad en base al Sistema Operativo de modo que tu le puedes asignar accesos de lectura, escritura y ejecución a cada columna, a cada table en base a los directorios y a los archivos que las contienen. Esto solamente se puede por esta otra razón, porque en VStar es “one process per query”, viene un query y el query al ejecutarse, autentifica, crea el proceso, se ejecuta el query y se muere el proceso. Eso tiene implicaciones por muchos lados, la primera es que esa es

la única manera de poder aprovechar la seguridad del Sistema Operativo. Segundo, ya te imaginarás, con respecto a lo de la distribución, quiere decir que en realidad a VStar no le importa donde esté corriendo el proceso, siempre y cuando tenga acceso, a través del SAN, (en el caso del SAN), a los archivos que son los que él espera que contengan la información. Ahora, un punto muy interesante, en cuanto a un process per query, es que no hay procesos corriendo. Entonces somos extremadamente light, muy ligeros, de tal manera, si no tienes queries ejecutando en el momento, somos Zero Foot Print. No hay ninguna huella de uso de la Base de Datos, cuando no se está ejecutando ningún query. Esto es muy diferente a Vertica, Vertica además tiene los dos espacios de almacenamiento, donde va pasando poco a poco la información de un lado a otro. Nosotros le hemos ganado a Vertica, de hecho le ganamos a Vertica en el Banco y el caso está que Vertica tiene cosas muy buenas, de hecho hace algunos años estuvieron hacienda mucho énfasis en la velocidad de carga, y la velocidad de carga que nosotros logramos con discos en paralelo, es prácticamente paralela a la de ellos, es una base de datos muy rápida. Pero, en otras cosas, no los llevamos. Por ejemplo, en el caso de VStar, en cuestiones de simplicidad de uso, por ejemplo, nosotros funcionamos a través de HTTP en cuanto a la manera de conectarte a la base de datos, quiero decir, puedes usar HTTP o HTTPS pero no hay una librería binaria. La hemos hecho antes para una conexión directa con Java, pero en realidad, el uso Standard no lo requiere. Entonces, tu query viene y es el Web Server el que recibe tu query, y el Web Server en base a la terminación nota que es un query para VSTAR y entonces ahí determina dónde va a disparar el proceso. Y fíjate que entonces nosotros obtenemos Load Balancing for free, porque el Load Balancing lo hace el Web Server, es decir, tu estableces el Web Server con carga balanceada en 3 nodos sin saber nada de VStar, si estás usando Apache, haces las técnicas de Apache. Nosotros recomendamos en Jinx, pero el caso es que es indistinto. Además enfrente de N Jinx tu pudiste haber tenido un cache como Squid, y para VStar es transparente.

G: Pues sí, son conexiones HTTP nadamás

E:Exacto. Entonces empiezas a entender cómo va la arquitectura. Con respecto al memory mapping es muy importante que no vaya a quedar una idea como que estamos hablando de algo en memoria, pore ej. Oracle Timestamp, es otra cosa, en realidad es una

de las cosas que la gente que ha usado memory mapping no se explica el porqué la demás gente no lo usa.
G: Exacto
E: La única explicación racional es que simplemente no se los enseñaron en la Universidad. Porque yo he visto gente que tiene que leer un archivo gigante de XML, más de 2 G, y se ponen a leerlo en Buffer
G: Si claro, lo cargan todo en memoria antes de leerlo.
E: Cuando tu puedes simplemente con Memory Mapping, mappeas y ya te descpreocupas, el sistema operativo se está encargando de traer de disco a memoria. Otra de las cosas relevantes que tiene arquitectura, nosotros no damos ninguna instrucción, no hay nada en el manual, por ejemplo hacer backups, porque nuestras estructuras de la Base de Datos, son estructuras normales del sistema operative, quiere decir que tú puedes utilizar culquier software que funcione en el sistema operativo para hacer backups en línea o en frío. Por ejemplo, para agregar una columna, nadamás agarras una columna, eso es lo que hacemos en la demo, cuando te hagan una demo, por ejemplo, tú tienes una base de datos con 250 millones , para darte de una vez ideas, nuestros límites están, en las bases de datos columnares, lo que importa es la cardinalidad, no el tamaño de la tabla para nada, es irrelevante. La cardinalidad límite, para nosotros, si quieres usar una sola máquina, nosotros favorecemos una máquina estilo Pizza box con doble procesador, no me refiero con Core, sino que tenga procesadores separados, de preferencia con Umba..con un arreglo de discos chicos, pero varios, o sea, es mucho mejor tener 6, 8 discos, que tener un discote. Cuando tienes este Pizza box, aprox un promedio de 1500 millones, dependiendo del estilo de queries, es lo realista que puedes esperar toparte si le pusiste 64 GB de RAM, a la Pizza Box y tienes dos procesadores rápidos. Si tienes 32 GB estás hablando de un límite por Pizza Box alrededor de 750 millones ó 1000. Con 16 GB de RAM, 250 millones es una cantidad razonable.. Pero mi punto original aquí es que tú acabas de hacer un query, agarras una columna, la quitas de la table y pueden seguir llegando queries a esa tabla y no pasa nada. Si llega un query, y hay una columna referenciada que no está en la tabla, simplemente el valor regresa null. No truena, no dice nada.

Igual tú puedes agregar una columna y la operación es inmediata, en cualquier otra base de datos, por ejemplo Vertica, el agregar una columna es una cosa que requiere su tiempo.
G: Ah sí, en Vertica se pueden agregar y es un rollote y luego no se pueden borrar.
E: Mira, otra cosa muy interesante resultado del diseño de la arquitectura es esta: sime dijeras, Ernesto..¿cómo particionas? ¿cómo haces un Shard?Pensemos primero en una partición vertical, tienes una tabla y quieres las primeras 3 columnas en un disco y las otras en otro disco, igual, no hay nada que saber específico de VStar, volvemos, lo único que le interesa saber a VStar es que el directorio que representa la table, le parezca al directorio que tienen las columnas adentro de . Y ahí ya te di la respuesta, simplemente creas links de las otras 3 columnas en el otro donde sea que esté a la tabla. A donde voy aquí es que es sumamente sencillo crear un Sharding vertical. Y hacerlo a otra máquina es igual, porque es lo mismo hacerlo de un file a system a otro en Linuz y de una máquina a otra. Finalmente para terminar la introducción, lo que mencionó Eduardo con respect al SAN, es una cosa sumamente interesante, la carga de datos a VStar, simplemente quiere decir que los datos aparezcan en forma binaria en una estructura de archive, que Vstar entienda como columna y tenga una referencia a ellos. Eso quiere decir muchas cosas, por ejemplo, que una columna misma puede estar compartida en muchas tablas, quiere decir, por ejmeplo, imagínate que estás cargando una enorme cantidad de datos a un disco y tu gente (los usuarios) están accesando el otro disco en tu server. En el momento en que terminas de cargar los nuevos datos, en ese momento, marcas el disco en el SAN, como disponible, y en ese momento la información está disponible para todas las workstations que estén conectadas a ese SAN. A diferencia, de que si fuera Oracle, o fuera Vertica, apenas habrías empezado, porque tendrías que mover los datos al lugar donde está la Base de Datos. Esta es una diferencia muy importante. La razón, es, que mientras tengas un Block Acces, no funciona con un NFS o con NAS, pero con un SAN sí, lo que necesitas es que sea Block Level.
G: No pues muy interesante. Una pregunta, ¿ustedes hacen algún tipo de compresión en las columnas? Por ejemplo si hay baja cardinalidad, los archivos son muy chicos supongo.
E: Mira, Gran parte del performance de Vertica, al principio, hace dos años que no studio Vertica, pero la situación de Vertica, la impression que me llevé es que VErtica deriva

gran parte de su performance a cuestiones de compresión, nosotros no. Nuestro performance viene de la sinergia del Memory Mapping, el hecho que tenemos un lenguaje abajo que es de arreglos, endemoniadamente rápidos y de que somos extremadamente ligeros. Esa es la razón principal de nuestro performance.
Nosotros no dependemos de la compresión para tener el performance. En las demos que hacemos no hacemos ningún tipo de compresión. Por ejemplo, el otro cliente importante que tenemos es el retail. Con retail están los casos que están los colores, marcas, y en lugar de guarder los valores como strings, nosotros los guardamos con lo que se llama un símbolo. Un símbolo tiene la ventaja de que se ve como string, o sea tu no ves 1, en vez de café o verde, tú ves café y verde, pero es un símbolo, no es el string café, me explico? Entonces el resultado es que tus queries se ven mucho más naturales, porque tu sabes cuando estás usando numerous para codificar ahorras espacio, pero el query dice color = 1 en vez de color = café. Ahí está un tipo de compresión, la compresión basada en el uso de tipos de datos que llevan ya su manera compacta de representarse. VStar lo hace autommáticamente, tu marcas una columna como simbólica y el contenido de string se preserva y se crea automáticamente una nueva columna…
G: Es un diccionario.
E: Exactamente. Pero se crea automáticamente una columna para tipo, el valor de símbolo es igual a la serie de caracteres, pero que solamente ocupa un byte, si son menos de 255 valores, o 2 bytes si son 16000, etc. Este tipo de compresión si la hacemos y se hace al natural. Ahora, nosotros podemos manejar texto con suma facilidad pero nosotros no hacemos nada de texto. La estrategia que hemos seguido e sue major integramos el software. Por ejemplo, uno de los software que siempre integramos es R, el software abierto de Estadísticas. R nos da a nosotros una base muy amplia de dif métodos para hacer modelaje estadístico, el punto aquí es que integramos biométricas. Por ejemplo los trabajos que hemos hecho para la policía federal, una de nuestra grandes ventajas, fue la manera en que integramos las biomérticas. Nosotros no hacemos algoritmos de reconocimiento facial ni de reconocimiento de huella, pero siempre hemos vendido VStar en esos ambientes. ¿Cómo le hemos hecho? Por ejemplo, siempre hacemos un partnership con una compañía, siempre es una compañía israelí y que luego se convirtió en norteamericana. Y ellos te venden el algoritmo que te hace el match de dos caras o dos

huellas. Y normalmente es una rutina en Set, pero está hecha para funcionar Stand Alone. Con cualquier otra base de datos, incluyendo Vertica, sit u tienes las huellas guardadas en Vertica o en Oracle, tu tienes que sacar tus huellas, meterlas en memoria, porque el algoritmo este solamente funciona así y corer el algoritmo. Imagínate que relajo.
G: Imagínate si es una base de datos gigante.
E: Exacto. Es más, ya con 20 millones ya te metiste en problemas. VStar no. Lo que nosotros hacemos regularmente, es hacerle creer al software de ellos que estamos corriendo en memoria, y hacemos un puente a la librería de C y esa librería ya la corremos sobre cada valor que le vamos pasando como argumento en una iteración. Entonces escencialmente no tenemos que mover los valores fuera de nuestras columnas para tener el resultado de los algoritmos y eso, así como te comento con las biometricas, lo podemos hacer con cualquier rutina de C. En particular, si ustedes tienen código en C++, nosotros favorecemos C, pero muchas cosas están escritas en C++. El caso es que cualquier rutina de C o C++ que tengas VStar la puede corer en el server. Con esta diferencia importante de SQL server y de Vertica. Porque Vertica también está escrita en C y puedes incorporar código, sin embargo nunca vi como le hicieron. Pero conozco los casos de Oracle y SQL server y el problema con estas es que si tienes error en tu rutina, o truena tu rutina, o te traes abajo el server, o sea la Base de datos. Porque estás corriendo en el proceso de la Base de Datos. VStar no. En VStar justo cuando llega el query al Web Server y en base a tu identidad, primero busca en tu directorio el archive ejecutable de VStar, entonces ahí tu puedes tener una versión de Vstar diferente, que es la que tendría embebido el código de C, por ejemplo, entonces se corre esa, en vez de correr la de más arriba, que es la que aplica para todos los usuarios, en caso de que no tengan uno especial, entonces se dispara esa. En caso de que tu query truene, nadie se enteró.
G: Perfecto