Artículo
Nancy Martínez · 8 oct, 2019 Lectura de 7 min

¿Cómo se cuentan las estrellas? Cómo se utiliza InterSystems Caché eXTreme en Gaia

Las herramientas que utilizan los astrónomos

Hace 6 años, el 19 de diciembre del 2013, la Agencia Espacial Europea (ESA) lanzó un telescopio orbital llamado Gaia. Podéis obtener más información sobre la misión Gaia en la página web oficial de la Agencia Espacial Europea o en el artículo de Vitaly Egorov (Billion pixels for a billion stars).

Y sin embargo, pocas personas saben cuál fue la tecnología que utilizó la agencia para almacenar y procesar los datos recopilados por Gaia. En 2011, dos años antes del lanzamiento, los desarrolladores barajaban varias opciones (consultar "Astrostatistics and Data Mining" escrito por Luis Manuel Sarro, Laurent Eyer, William O'Mullane, Joris De Ridder, pp. 111-112):

- *IBM DB2*,
- *PostgreSQL*,
- Apache Hadoop,
- Apache Cassandra and
- *InterSystems Caché* (para ser más precisos, la tecnología Caché eXTreme Event Persistence).

Al comparar las tecnologías entre sí, se obtuvieron los siguientes resultados (fuente):

Tecnología	Tiempo
DB2	13min55s
PostgreSQL 8	14min50s
PostgreSQL 9	6min50s
Hadoop	3min37s

Tecnología	Tiempo
Cassandra	3min37s
Caché	2min25s

Los primeros cuatro probablemente son conocidos incluso por los niños en edad escolar. Pero, ¿qué es Caché XEP?

Tecnologías Java en Caché

Si se observa el conjunto de APIs de Java proporcionado por InterSystems, veremos lo siguiente:

- La tecnología Caché Object Binding proyecta los datos de forma transparente en Java. En términos de Caché, las clases del proxy que se generaron en Java se llaman exactamente así, proyecciones. Este enfoque es el más sencillo ya que guarda las relaciones " naturales " entre las clases en el modelo de objetos, pero esto no garantiza que tengan un gran rendimiento: muchos de los metadatos del servicio que describen el objeto del modelo se transfieren " de forma específica "
- **J**DBC y varias extensiones (Hibernate, JPA). Supongo que hasta aquí no hay nada nuevo aparte del hecho de que Caché es compatible con dos tipos de aislamiento durante sus transacciones: READUNCOMMITTED y READCOMMITTED – y funciona de modo predeterminado con READUNCOMMITTED
- La familia *Caché eXTreme* (también está disponible en las ediciones de .NET y Node.js). Este enfoque se caracteriza por el acceso directo a la representación de los datos de bajo nivel (los llamados "globales", es decir, la cantidad de datos en el mundo de Caché) lo que garantiza un alto rendimiento. La librería del XEP que utiliza Caché proporciona acceso simultáneo a los objetos y a los datos cuasi relacionales.
    - Objeto - La API del cliente ya no necesita preocuparse por las representaciones objeto-relacional: siguiendo el modelo de objetos en Java (incluso en los casos de herencia multinivel compleja), el sistema crea automáticamente un modelo de objetos que están al nivel de las clases de Caché (o un esquema de base de datos si queremos utilizar los términos para la representación del relacional)
    - Cuasi-relacional, se refiere a que es posible ejecutar consultas SQL en diversos " eventos " almacenados en una base de datos (para ser exactos, las solicitudes que utilizan el subconjunto SQL) directamente desde el contexto de una conexión eXTreme. Además, los índices y las transacciones son totalmente compatibles. Por supuesto, es posible tener acceso inmediato a todos los datos que se cargaron y a una representación del relacional mediante JDBC (el cual es compatible con todas las poderosas funciones de las extensiones ANSI SQL y SQL específicas para el lenguaje de Caché), pero la velocidad de acceso será completamente diferente.

En resumen, esto es lo que tenemos:

- La importación del " esquema " (en Caché las clases se crean automáticamente), que incluye:
- la importación de la jerarquía de las clases en Java,
- acceso instantáneo relacional a los datos: puede trabajar con las clases de Caché de la misma manera que trabaja con las tablas,
- es compatible con los índices y las transacciones mediante Caché eXTreme,
- es compatible con las consultas SQL simples mediante Caché eXTreme,
- es compatible con las consultas SQL arbitrarias mediante la conexión JDBC que subyace respecto al TCP (Caché utiliza el controlador estándar Tipo 4, el cual funciona directamente con la base de datos Pure de Java).

Este enfoque ofrece algunas ventajas en comparación con relacionales similares (mayor velocidad de acceso) y varias soluciones NoSQL (acceso instantáneo a los datos de tipo de relacional).

La " peculiaridad " en la configuración de Caché eXTreme es que antes de la conexión debe configurarse el entorno:

- la variable GLOBALSHOME debe apuntar a la carpeta de instalación en Caché y
- LDLIBRARYPATH (DYLDLIBRARYPATH para *Mac OS X* o PATH para *Windows*) debe contener ${GLOBALSHOME}/bin.

Además, es posible que se requiera aumentar los tamaños de stack y heap en JVM (-Xss2m -Xmx768m).

Un poco de práctica

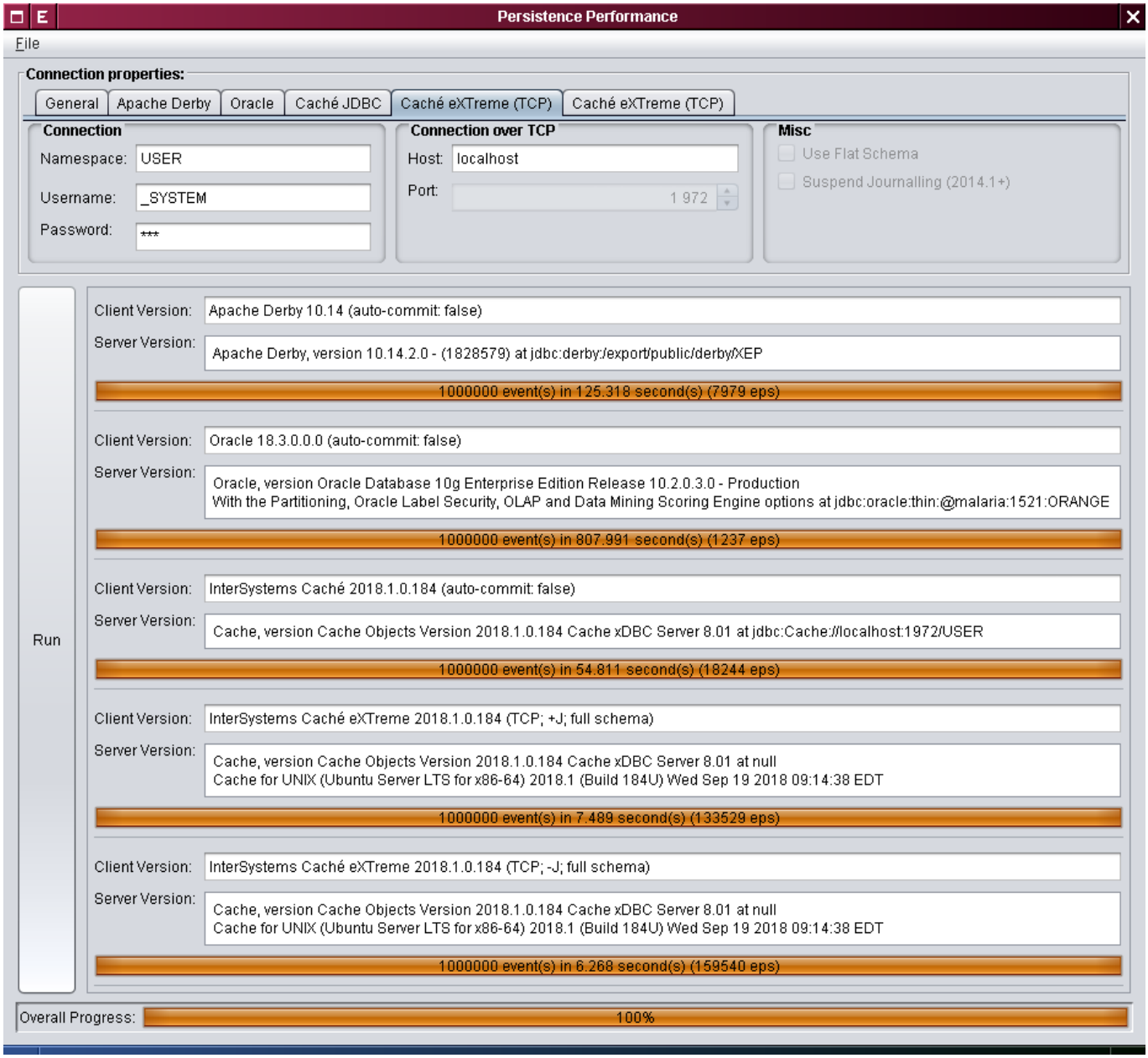
Los autores estaban interesados en saber cómo se comportaría Caché eXTreme mientras se escribe un flujo continuo de datos, en comparación con otras tecnologías de procesamiento de datos. Utilizamos datos históricos de cotizaciones bursátiles en formato CSV, los cuales se encuentran disponibles en el sitio web de la organización de empresas financieras "Finam" . Lo siguiente es una muestra del archivo de datos:

```
<TICKER>,<PER>,<DATE>,<TIME>,<LAST>,<VOL>
NASDAQ100,0,20130802,09:31:07,3 125.300000000,0
NASDAQ100,0,20130802,09:32:08,3 122.860000000,806 906
NASDAQ100,0,20130802,09:33:09,3 123.920000000,637 360
NASDAQ100,0,20130802,09:34:10,3 124.090000000,421 928
NASDAQ100,0,20130802,09:35:11,3 125.180000000,681 585
```

El código de la clase modelada en Caché para la estructura anterior podría verse así:

```
Class com.intersystems.persistence.objbinding.Event Extends %Persistent [ ClassType =
 persistent, DdlAllowed, Final, SqlTableName = Event ]
{
Property Ticker As %String(MAXLEN = 32);
Property Per As %Integer(MAXVAL = 2147483647, MINVAL = -2147483648);
Property TimeStamp As %TimeStamp;
Property Last As %Double;
Property Vol As %Integer(MAXVAL = 9223372036854775807, MINVAL = -9223372036854775810)
;
}
```

También escribimos un poco de código de prueba básico y simple. Este enfoque "simple" puede justificarse por el hecho de que en realidad no estamos midiendo la velocidad del código generado por JIT, sino la velocidad a la que el código, que no tiene ninguna relación con el JVM (con excepción de Apache Derby), puede escribir en el disco. Así es como se verá la ventana del programa de prueba:



Nuestra competencia es:

- *Apache Derby* 10.14.2.0
- *Oracle* 10.2.0.3.0
- *InterSystems Caché* 2018.1 (JDBC)
- *InterSystems Caché* 2018.1 (eXTreme)

Dado que las pruebas son un tanto aproximadas, no vimos ningún propósito práctico en proporcionar las cifras exactas: el margen de error es bastante alto, mientras que el objetivo del artículo es demostrar la tendencia general. Por las mismas razones, no especificamos la versión exacta de JDK y la configuración del recolector de basura (Garbage Collector): el lado del servidor JVM 8u191 con -Xmx2048m -Xss128m alcanzó un nivel de rendimiento muy similar en Linux y Mac OS X. En cada prueba se guardaron un millón de eventos, se realizaron varias ejecuciones en caliente (hasta 10) antes de cada prueba para una base de datos en particular. En cuanto a la configuración de Caché, la rutina de caché aumentó a 256 MB y la base de datos en caché de 8KB incrementó hasta 1024 MB.

Nuestras pruebas arrojaron los siguientes resultados (los valores de la velocidad de escritura se expresan en eventos por segundo (eps)):

Tecnología	Tiempo, s (menos es mejor)	Velocidad de escritura, eps (más es mejor)
Apache Derby	140±30	7100±1300
Oracle	780±50	1290±80
Caché JDBC	61±8	17000±2000
Caché eXTreme	6.7±0.8	152000±17000
Caché eXTreme, transaction journaling disabled	6.3±0.6	162000±14000

1. Derby ofrece velocidades que varían desde 6200 hasta 8000 eps
2. Oracle resultó tener una rapidez de hasta 1290 eps
3. Caché en el modo JDBC proporciona una mayor velocidad (desde 15000 hasta 18000 eps), pero hay una disyuntiva: el nivel de aislamiento de la transacción predeterminado, como se mencionó anteriormente, es READUNCOMMITTED
4. La siguiente opción, Caché eXTreme, nos da desde 127000 hasta 167000 eps
5. Por último, nos arriesgamos un poco y desactivamos el registro de las transacciones (para un proceso determinado por el cliente) y logramos alcanzar la velocidad de escritura de 172000 eps en un sistema de prueba

Quienes estén interesados en obtener números más precisos pueden consultar el código fuente. Será necesario lo siguiente para construirlo y ejecutarlo:

- *JDK* 1.8+,
- *Git*,
- *Maven*
- *Oracle* JDBC driver (disponible desde Oracle Maven Repository)
- Maven Install Plugin para crear un *Caché JDBC* local y los artefactos de *Caché eXTreme*:

```
$ mvn install:install-file -Dfile=cache-db-2.0.0.jar
$ mvn install:install-file -Dfile=cache-extreme-2.0.0.jar
$ mvn install:install-file -Dfile=cache-gateway-2.0.0.jar
$ mvn install:install-file -Dfile=cache-jdbc-2.0.0.jar
```

y, por último,

- *Caché* 2018.1+.

#Bases de datos #Java #JDBC #Prueba #Caché

URL de fuente:https://es.community.intersystems.com/post/%C2%BFc%C3%B3mo-se-cuentan-las-estrellas-c%C3%B3mo-se-utiliza-intersystems-cach%C3%A9-extreme-en-gaia