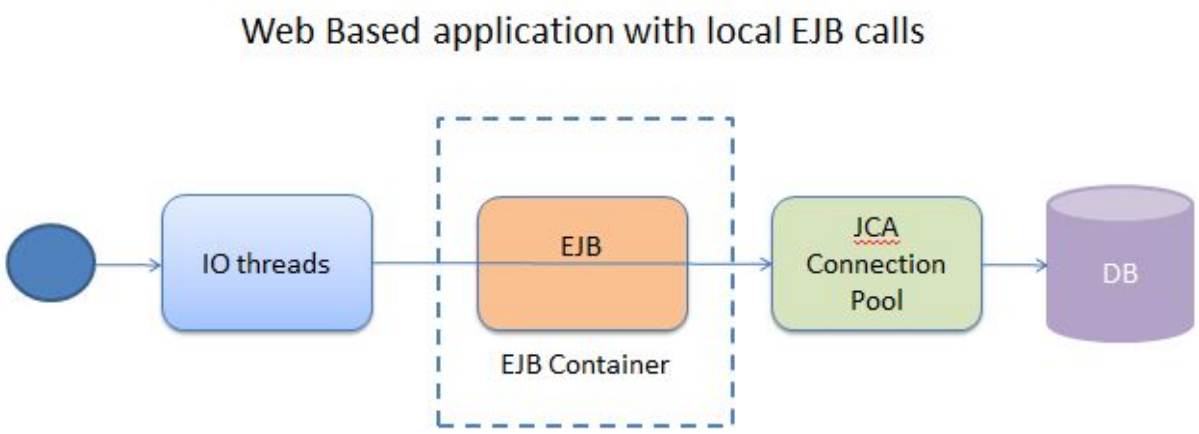
WildFly performance tuning

Nos centraremos ahora en **request flows** entrantes a medida que cruzan las fronteras Wildfly. Y vamos a aprender a controlar los atributos más importantes.

Si desea afinar una herramienta compleja como un servidor de aplicaciones Java, es crucial que usted entiende los pasos individuales que realiza la solicitud antes de llegar al componente de destino. Todo el resto tiene por lo general un impacto mínimo en el rendimiento.

Así que vamos a ver una petición típica procedente de un front-end web y llegar a JBoss EJB que a su vez invoca la base de datos:



Como se puede ver en la imagen superior, el hilo que está conduciendo a una solicitud basada en la Web es **Undertow XNIO hilo** que se crea por manipuladores de resaca. Ajuste de la cantidad correcta de **IO threads** es crucial en este escenario, de lo contrario tendrá un cuello de botella en el comienzo de su solicitud.

Siempre que existan suficientes **io-threads** para servir a su petición http, el **core-max-threads** (primero) y la **task-max-threads** (después) se utilizan para determinar en la solicitud se sirve o si va a ser desechada .

Hilos de resaca se configuran a través del **IO.** subsistema Aquí es cómo establecer el número XNIO piscina para un sitio web de tamaño medio-grande:

```
/subsystem=io/worker=default/:write-attribute(name=task-core-threads,value=25)
/subsystem=io/worker=default/:write-attribute(name=task-max-threads,value=100)
/subsystem=io/worker=default/:write-attribute(name=io-threads,value=100)
```

Aun en la capa web, es necesario tener en cuenta el **number of sessions** que están en ejecución. Los parámetros **max-active-sessions** se utiliza para determinar cuántos se le permitirá sesión HTTP activo. Si la creación sesión haría que el número de sesiones activas para superar <max-active-sessions/>, la sesión más antigua conocida por el gestor de sesiones se **pasivar** para hacer espacio para la nueva sesión. Eso es un gran precio en términos de rendimiento, por lo que vamos a ver cómo configurarlo usando jboss-web.xml:

```
<jboss-web>
   <max-active-sessions>200</max-active-sessions>
</jboss-web>
```

A continuación, , mantener el control del número de sesiones utilizando la CLI. Con el fin de hacer eso, es necesario que apunte a la **/ deployment** ruta y entrar en su **undertow:**

```
/deployment=webapp.war/subsystem=undertow/:read-resource(include-runtime=true)
{
        "outcome" => "success",
        "result" => {
            "active-sessions" => 1,
            "context-root" => "/webapp",
            "server" => "default-server",
            "sessions-created" => 1,
            "virtual-host" => "default-host",
            "servlet" => {
                "Faces Servlet" => undefined,
                "demo.DownloadServlet" => undefined
            }
        }
}
```

En el EJB Container

Una vez que el IO-thread alcanza el contenedor EJB, y EJB es recogido de la **Pool** (Stateless EJB) o inmovilizados de la **Cache** (Stateful EJB). Este es el segundo elemento que debe tener en cuenta en este escenario: que tiene suficientes recursos en su contenedor EJB.

Como cuestión de hecho, si está utilizando Beans de sesión sin estado que debe ser consciente de que **la puesta en común de EJB no está activado por defecto en WildFly 8**.El fundamento de esta decisión es que un tamaño exacto de la piscina es fuertemente dependiente de la aplicación y no se puede adivinar fácilmente. Por lo tanto, una piscina mal configurado de EJB podría ser incluso perjudicial en términos de rendimiento, provocando ciclos excesivos de recogida de basuras. Por estas razones, le toca a usted para activar esta función y definir una configuración correcta para sus parámetros.

Esto requiere sin embargo tan poco como una selección de cuadro combinado de la pestaña EJB3 de contenedores. O un simple comando CLI:

```
/subsystem=ejb3/:write-attribute(name=default-slsb-instance-pool,value=slsb-strict-
max-pool)
```

Si está utilizando **Stateful EJB** que más bien va a tratar con un caché de Beans que

contienen datos de conversación.

La memoria caché con estado por defecto utiliza un sistema de almacenamiento en caché sencilla que la *no* implica pasivación de los datos. Esto podría ser deseable en términos de rendimiento, sin embargo, si es necesario utilizar un mecanismo de pasivación en su aplicación, debe aprender a configurar los límites de la memoria caché.

Al principio es necesario tener la SFSB utilizar una pasivación caché capaces como el distribuibles:

```
/subsystem=ejb3/:write-attribute(name=default-sfsb-cache,value=distributable)
```

A continuación, puede configurar el número máximo de SFSB permitido en la memoria caché. Por ejemplo, la caché distribuible utiliza laInfinispan **pasivación-store** que permite que un número máximo de 10000 elementos de la cache

```
subsystem=ejb3/passivation-store=infinispan/:write-attribute(name=max-size,value=10
000)
```

Monitorizando el contenedor EJB

Una vez que ha configurado su piscina o caché, es el momento para controlar si la configuración es la adecuada. En cuanto a la capa Web, puede obtener esta información a través de su **despliegue,** unidad de cavando en el **ejb3** subsistemade la siguiente manera:

```
/deployment=myapp.jar/subsystem=ejb3/stateless-session-bean=Manager/:read-resource(
recursive=false,include-runtime=true,include-defaults=false)
{
        "outcome" => "success",
        "result" => {
            "component-class-name" => "Manager",
            "declared-roles" => [],
            "execution-time" => 0L,
            "invocations" => 0L,
            "methods" => {},
            "peak-concurrent-invocations" => 0L,
            "pool-available-count" => 20,
            "pool-create-count" => 1,
            "pool-current-size" => 1,
            "pool-max-size" => 20,
            "pool-name" => "slsb-strict-max-pool",
            "pool-remove-count" => 0,
            "run-as-role" => undefined,
            "security-domain" => "other",
            "timers" => [],
            "wait-time" => 0L,
            "service" => undefined
        }
}
```

Dentro de EIS

Si usted tiene suficientes recursos en el contenedor EJB, entonces el proceso de IO continuará su carrera hasta el último paso que suele ser la base de datos, pero podría ser también otra EIS.

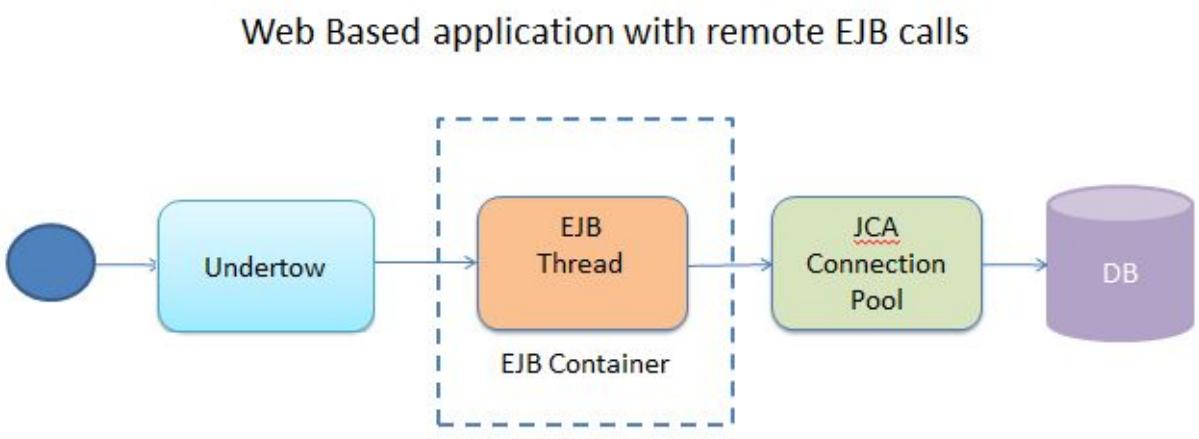
En caso de que se trata de conexiones de bases de datos, debe adquirir una conexión de la pool, que se rige por la capa JCA. El parámetro de configuración clave es max-pool-size que especifica el número máximo de conexiones para un poll. (Por defecto 20). Tenga en cuenta que habrá un límite máximo para el número de conexiones permitidas por la base de datos para que coincida.

Usted puede configurar su **Connection pool max size** de un atributo diferente utilizando esta CLI:

```
/subsystem=datasources/data-source=MySQLDS/:write-attribute(name=max-pool-size,value=50)
```

La EJB solicitud de flujo remoto

En el segundo ejemplo, que cubrirá un enfoque ligeramente diferente que implica llamadas EJB remotos. (Piense por ejemplo de un cliente EJB remoto).



Desde Wildfly 8, estas llamadas no están aterrizando directamente en el contenedor EJB, pero que están mediados por Undertow que lleva a cabo un **Http upgrade** hacia el canal de comunicación remota.

Como se puede adivinar a partir de la imagen, el proceso de IO no es más que el conductor de la llamada, pero el **conjunto de subprocesos EJB** entra en juego. Puede configurar la agrupación de hebras de EJB desde el subsistema ejb3 de la siguiente manera:

```
/subsystem=ejb3/thread-pool=default/:write-attribute(name=max-threads,value=100)
```

En este ejemplo, hemos establecido a 100 que debe ser un valor bastante alto.

Los hilos Undertow todavía tienen alguna relevancia en este escenario, ya que se pueden utilizar en caso de que utilice **async** las llamadas a su EJB y para proporcionar la **response** al cliente.

Puede supervisar el grupo de subprocesos en tiempo de ejecución utilizando la CLI siguiente:

```
/subsystem=ejb3/thread-pool=default/:read-resource(include-runtime=true)
{
        "outcome" => "success",
        "result" => {
            "active-count" => 5,
            "completed-task-count" => 13L,
            "current-thread-count" => 3,
            "keepalive-time" => {
                "time" => 100L,
                "unit" => "MILLISECONDS"
            },
            "largest-thread-count" => 5,
            "max-threads" => 10,
            "name" => "default",
            "queue-size" => 0,
            "rejected-count" => 0,
            "task-count" => 0L,
            "thread-factory" => undefined
        }
}
```