Trabajo de Investigación: Integración HPC con proyecto de MyPetFeeder

<<Brude Alejandro, Castillo Tomas, Fernandez Julian, Fernandez Nicolas, Orlando Javier, Vargas Gabriel>>

[1]Universidad Nacional de La Matanza,
Departamento de Ingeniería e Investigaciones Tecnológicas,
Florencio Varela 1903 - San Justo, Argentina
<<alejandrobrude@gmail.com, castillotomas96@gmail.com, juliangonzalofernandez@gmail.com, ni.ko94@hotmail.com, javierorlando@ymail.com, gabilol.soad@gmail.com>>

Resumen. En el siguiente informe se van a explicar brevemente los beneficios del HPC sobre el proyecto de IoT MyPetFeeder.

Tal proyecto consiste en facilitar la alimentación de mascotas de un usuario común y corriente. El usuario solo debe tener cargado el dispositivo con alimento para la mascota y luego se irán tomando estadísticas para informarle al usuario el estado de cómo se está alimentando su perro.

El objetivo de este informe es sobre como implementar HPC utilizando MPI. Los objetos para considerar serán, un servidor y los distintos dispositivos Arduino y Android. Cada uno sobre distintas razas de perros para obtener el mejor de promedio de alimentación posible para cada una.

Palabras claves: Arduino,Android,MPI,HPC.

1 Introducción

MyPetFeeder es un sistema embebido conformado por un contenedor de alimentos y un servo motor conectado a un arduino con varios sensores para dispender los alimentos al animal en tiempo y forma. Luego de cada alimentación el embebido toma información sobre el proceso de alimentación y con dicha información se toman estadísticas y se va informando al usuario distinto tipo de información como que su animal está consumiendo valores de comida fuera de los parámetros generales. También puede informar si su perro está consumiendo la comida en tiempo indebido.

Al manejarse con una aplicación en Android partimos de la idea de conectar todos los dispositivos Android e ir compartiendo la información entre los diferentes usuarios para luego centralizar todo en un servidor unico que utilizara como sistema operativo Windows server 2012 R2. Por medio del cual se establecerá la comunicación de los dispositivos android con el host para recibir en el mismo las estadísticas sobre los programas asociados a los datos de las mascotas esto servira

para analizar la BigData, producir estadísticas generales y así poder dar un feedback más desarollado y refinado asistido por la opinión experta y reajustar la rutina de alimentación en base a las estadísticas generales obtenidas

2 Desarrollo

Hoy en día, muchas personas tienen mascotas y no tienen conocimiento sobre la alimentación de las mismas. Dada la recolección de datos desde las asociaciones veterinarias de distintas partes del mundo, se han confeccionado tablas de referencia para la alimentación canina:

RACIÓN DIARIA (TAZAS/GRAMOS)		
PESO	GRAMOS	CANTIDAD
1 - 3 kg	50 - 90 g	1/2 a 1 taza
3 - 5 kg	90 - 120 g	1 a 1 1/4 taza
5 - 10 kg	120 - 190 g	1 1/4 a 1 3/4 taza
10 - 15 kg	190 - 260 g	1 3/4 a 2 1/2 taza
15 - 20 kg	260 - 310 g	2 1/2 a 3 taza
20 - 30 kg	310 - 410 g	3 a 4 taza
30 - 40 kg	410 - 500 g	4 a 4 3/4 taza
40 - 50 kg	500 - 590 g	4 3/4 a 5 1/2 taza
50 - 60 kg	590 - 660 g	5 1/2 a 6 1/4 taza
60 - 70 kg	660 - 740 g	6 1/4 a 7 taza
70 - 80 kg	740 - 800 g	7 a 7 1/2 taza

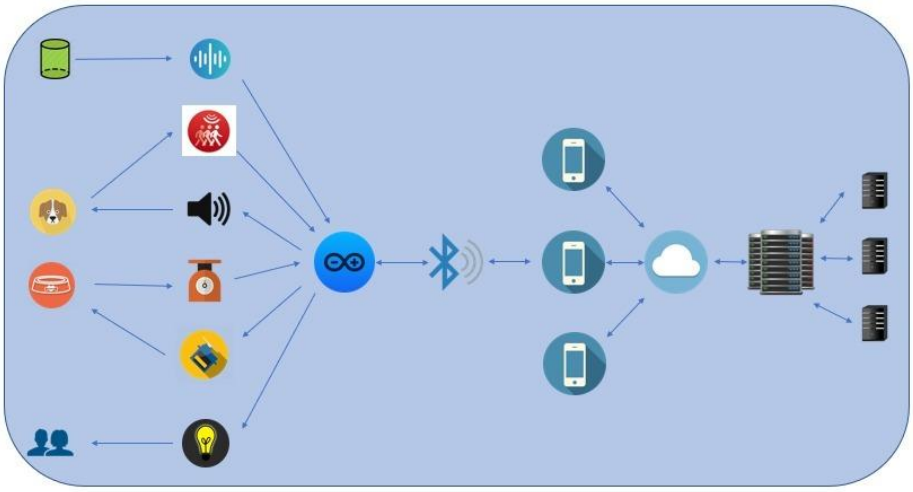
Sin embargo, en el intento de ir más profundamente, en el afán de mejorar la calidad de vida de nuestras mascotas, se ha propuesto mejorar el rendimiento de los argoritmos, y refinar aun más los valores por medio del procesamiento masivo de datos. Se ha decidido implementar tecnología OpenMPI para el procesamiento de la información en paralelo, en sinergia con las tecnologías ya implementadas de android y arduino, especificado esto, se determinó una infrasectructura mediante la cual la terminal Android se comunicará por medio de WebService con los servidores centralizados distribuidos alrededor del globo, lo cual permite el analisis de las estadísticas locales, con un enfoque mas generalizado, cada Data center cuenta con la tecnología adecuada para implementar procesamiento de datos distribuidos, entre los distintos nodos locales, los cuales asisten de manera coordinada con el fin de pulir los inputs, y ajustar de la mejor manera los valores de inyección alimenticia, lo que asegura una mejor salud de nuestros compañeros.

Sabiendo gracias a esto que el uso de HPC puede brindar mayor precisión en la lectura de valores, la investigación propuesta para nuestro embebido, continuará su desarrollo sobre cómo dividir los clúster (o sea, el conjunto de computadoras unidas mediante una red de alta velocidad funcionando como una única computadora) quienes son fundamentales a la hora de maximizar la eficiencia de ejecución en paralelo. Esto sería utilizar el middleware HPC Java, o también

llamado JCL5, el cual permite trabajar con dispositivos tanto Android como Java de manera asíncrona específica para IoT. Para la implementación de MPI en dispositivos móviles Android utlilizaremos la biblioteca MPICH2 que permitirá armar los clúster portables con dispositivos que usen el sistema operativo Android armar los clúster portables con dispositivos que usen el sistema operativo Android.

3 Explicación del algoritmo.

Para desarrollar la solución propuesta debemos implementar el servidor (con los distintos clústers) conectado a los múltiples dispositivos Android que recopilarán los datos de alimentación de sus respectivos alimentadores conectados y a través de MPI se procesaran los datos leídos para obtener información más precisa sobre la alimentación de cada mascota. Cabe aclarar que la conexión entre Android y el servidor se realiza por medio de un WebService (según lo investigado TomCat Server sería una muy buena opción) Esto queda representado en el siguiente diagrama:



El servidor, con toda la información recopilada procederá a un análisis de los datos recibidos de la alimentación para poder brindar posteriormente a los usuarios la información del estado de alimentación de su mascota.

El funcionamiento básico de ambas partes (servidor y android) sería algo similar al siguiente pseudocódigo:

```
#Node de salida (Arduino)
#Arduino::Send(pesoBalanza,pesoDeposito,velocidadAlimentacion)
#
#Nodo de entrada (Android)
#Android::Receive(pesoBalanza,pesoDeposito,velocidadAlimentacion)
#
#Nodo de salida (Android)
#Android::Send(raza,peso,fechaNaciemiento,cantidadConsumida,cantidadInyectada,nivelActividadPerro)
#
#Nodo de entrada (NodoCentral)
#NodoCentral::Receive(raza,peso,fechaNaciemiento,cantidadConsumida,cantidadInyectada,nivelActividadPerro)
#
#Nodo de salida(NodoCentral)
#NodoCentral::mpi_send(raza,peso,fechaNaciemiento,cantidadConsumida,cantidadInyectada,nivelActividadPerro)
#
#Nodo de entrada(NodosSecundarios)
#NodosSecundarios::mpi_recv(raza,peso,fechaNaciemiento,cantidadConsumida,cantidadInyectada,nivelActividadPerro)
#NodosSecundarios::process(raza,peso,fechaNaciemiento,cantidadConsumida,cantidadInyectada,nivelActividadPerro)
#
#
#Nodo de entrada(NodoCentral)
#NodoCentral::mpi_recv(cantidadRecomendada,ingestasRecomendadas)
#cantRecAndroid = calcularCantidadRecGlobal(raza,peso,fechaNacimiento,cantidadRecomendada)
#cantIngAndroid = calcularCantidadIngGlobal(raza,peso,fechaNacimiento,ingestasRecomendadas)
#
#Nodo de salida(NodoCentral)
#NodoCentral::send(cantRecAndroid,cantIngAndroid)
#
#Nodo de entrada(Android)
#Android::Receive(cantRecAndroid,cantIngAndroid)
#Android::AjustarRutina = Android::Process(cantRecAndroid)
#Android::AjustarRutina = Android::Process(cantIngAndroid)
```

Nodo secundario

```
#Nodo de procesamiento (NodoSecundario)
#cantidadRecomendada = 0
#ingestasRecomendadas = 0
#NodosSecundarios::mpi_recv(raza,peso,fechaNaciemiento,cantidadConsumida,cantidadInyectada,nivelActividadPerro)
#coeficiente = nivelActividadPerro/peso
#anios = ConvertirAAnios(fechaNaciemiento)
#valorReferencia = Anios*Coeficiente*raza
#resto = CalcularResto(cantidadInyectada - cantidadConsumida)
#switch(resto){
#       case1::(valorReferencia*2 < resto){
#               cantidadRecomendada -= 2*valorReferencia
#               ingestasRecomendadas += 2
#       }
#       case2::(valorReferencia < resto <= valorReferencia*2){
#               ingestasRecomendadas += 1
#       }
#       case3::(0 < resto <= valorReferencia){
#               cantidadRecomendada -= valorReferencia
#       }
#       case4::(resto == 0){
#               cantidadRecomendada += valorReferencia
#               ingestasRecomendadas += 1
#       }
#}
#NodosSecundarios::mpi_send(cantidadRecomendada,ingestasRecomendadas)
```

4 Pruebas que pueden realizarse

Para probar la mejora basada en HPC, en el sistema alimentador de mascotas, la evaluación fundamental consistiría en ver si los parámetros de alimentación obtenidos e informados a los usuarios finales son acordes. Por ejemplo, si un perro de 5 kilos consume 1 kilo de alimente por día, la prueba sería totalmente fallida. Pero en caso de obtener un resultado mucho más lógico y acorde a los estándares generales de alimentación de perros, se puede afirmar que el proyecto de investigación con base HPC cumplió con su finalidad.

5 Conclusiones

En base a lo investigado y desarrollado, podemos afirmar que la propuesta para ampliar el alimentador de perros con HPC podría ser muy buena para aquellas personas que no le den importancia a la alimentación de sus perros y le evitarán problemas a largo plazo sobre los mismos.

En los tiempos que corren actualmente la dedicación a alimentación de las mascotas no es prioridad de las personas en general. Por esto mismo nuestra misión es mejorar la vida de nuestros pequeños caninos para que sus dueños los puedan disfrutar por el mayor tiempo posible.

Por ejemplo, una persona que trabaja 8/9 horas diarias pasa poco tiempo con su perro entonces es altamente probable que no le de importancia a la alimentación y esto pueda generar trastornos alimentacicios.

6 Referencias

1. Guamán Rivera; Brayme Lino: Análisis de rendimiento de un clúster HPC y arquitecturas manycore y multicorer. Tesis de grado. Cuenca. (2017)
2. Instituto de Ciencias Exatas e Biologicas; Universidade Federal de Ouro Preto ; Instituto Federal de Minas Gerais; Universidade Federal de Alagoas. JCL Android: Uma Extensao IoT para o Middleware HPC JCL. Investigación compartida entre dichas universidades. Brasil. (2017).
3. Tomcat: https://medium.com/@valerybriz/gu%C3%ADa-completa-para-montar-un-servlet-con-java-tomcat-y-ant-explicado-paso-a-paso-windows-b2ff203c50d3
4. Alimentación:https://www.hogarmania.com/mascotas/perros/alimentacion/201705/cantidad-comida-perros-35991.html