DISEÑO, IMPLEMENTACIÓN Y EVALUACIÓN DE CLÚSTER RB-PI PARA EL PROCESAMIENTO DE VISIÓN EN PARALELO

Muñoz-Saavedra, Luis; Domínguez-Morales, Manuel*; Cascado-Caballero, Daniel; Linares-Barranco, Alejandro

Grupo de Robótica y Tecnología de Computadores.
Departamento de Arquitectura y Tecnología de Computadores. Escuela Politécnica Superior. Universidad de Sevilla.

*E-mail: mjdominguez@us.es

RESUMEN

En este proyecto se presenta la creación de un clúster de micropcs para análisis de imágenes. Para su creación partimos de cero: el análisis de las herramientas que nos ayuden a conseguir el objetivo, un análisis de las prestaciones, así como los pasos para su creación. Se parte de la base de que se alcanzó el límite de procesamiento monoprocesador desde hace varias generaciones, lo cual ha supuesto una fuerte evolución en el procesamiento paralelo. Para aplicaciones con grandes necesidades (cálculos matemáticos, procesamiento de vídeo, entrenamiento, etc.) se hace uso de clústeres; sin embargo, éstos poseen un coste muy elevado, lo cual provoca que sea uso exclusivo en supercomputación. En este trabajo se presenta el diseño e implementación de clúster de bajo coste basado en computadores empotrados Raspberry Pi. Se describe la configuración del clúster para comunicación por paso de mensajes. Y, finalmente, se realiza una prueba del sistema utilizando aplicación de procesamiento paralelo de vídeo.

PALABRAS CLAVE

Clúster, OpenCV, Raspberry, Cisión artificial, Paralelismo.

ABSTRACT

This project presents the creation of a micro-pc cluster for image analysis. For its creation we start from scratch: the analysis of the tools that help us achieve the goal, an analysis of the benefits, as well as the steps for its creation. Since the limit of single-processor processing has been reached for several generations, a strong evolution in parallel processing has been experimented in the last years. For applications with great computation power needs (mathematical calculations, video processing, training, ...) clusters are used; however, they have a very high cost, which makes them to be used exclusively in supercomputing. In this paper we present the design and implementation of a low-cost cluster based on Raspberry Pi embedded computers. The configuration of the cluster for communication by message passing is described. And, finally, a system test is performed using a parallel video processing application.

KEYWORDS

Cluster, OpenCV, Raspberry, Computer vision, Parallelism.

INTRODUCCIÓN Y OBJETIVOS

Un campo en auge y que requiere gran capacidad computacional es la catalogación automática de imágenes. Puede apreciarse, por ejemplo, el proyecto Galaxy Zoo [1] que pretende catalogar imágenes de galaxias, lo cual no es un tema sencillo, ya que las imágenes no suelen ser fácilmente catalogables de manera informática debido al ruido que tienen estas fotos. Para evitar este problema, el programa usa voluntarios para catalogar estas imágenes. Con esta metodología surgen varias preguntas, ¿cuántas fotos puede catalogar una persona por minuto?, ¿qué pasa si no hay voluntarios?, y la más interesante, ¿se podría crear un sistema experto para que, mediante aprendizaje automático pudiese reconocer estas imágenes? ¿Cómo debería ser este sistema?

En la temática de este trabajo, se ha obviado la parte de aprendizaje automático, y se ha centrado en la parte arquitectónica del problema, ya que es importante definir correctamente la arquitectura del sistema necesario y probar que sea capaz de realizar las operaciones que se precisan para este tipo de iniciativas. Este trabajo se puede dividir en dos partes, una más general, los sistemas HPC basados en mini PCs, y uno más particular, un clúster HPC para análisis de imágenes.

Un clúster HPC formado por mini PCs podría usarse en estos campos: Infraestructuras, Educativo, Científico, etc. Y ya en un campo más específico: Sanitario, Seguridad, Automoción, etc.

Aunque, en la actualidad, se tiende a llevar los servicios software a la nube, para ello hay que externalizar, y normalmente se simplifican las tareas de mantenimiento, ya que la parte de mantenimiento de las máquinas las hace otra empresa, pero, para poder realizar las pruebas de código, estos clústeres pueden ahorrar dinero y tiempo, ya que normalmente este tipo de servicios tarifica por el tiempo de procesado.

METODOLOGÍA

El sistema que se va a construir es el siguiente:

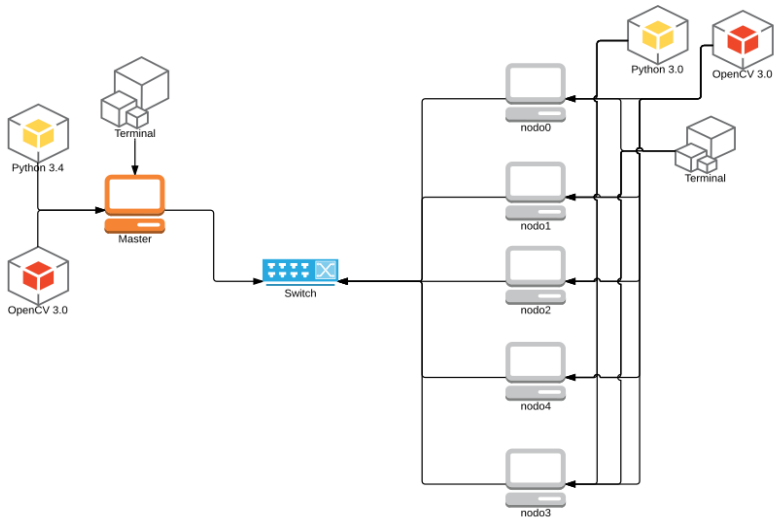

Figura 1. Sistema basado en clúster con un maestro y 5 nodos esclavos.

Como vemos en el dibujo, el sistema está separado en Máster y nodos, aunque tenga una configuración similar, el máster tiene una configuración algo diferente.

- **Máster:** El máster hace uso del sistema operativo completo de Raspbian, es decir cuenta con entorno gráfico, se ha hecho así para poder usar esta Rasberry como entorno de desarrollo en caso de que fuese necesario, ya que incluye un IDE para Python, está Raspberry también ha sido configurada para poder conectarse mediante SSH al resto de nodos, ya que tiene clave pública distribuida entre los nodos.

- **Nodos:** Los nodos, tienen una versión mínima del sistema operativo, ya que están pensadas solo para realizar trabajos. En cuanto al software usado para el procesamiento de las imágenes, es igual en todas las máquinas, se usa Python 3.4 y OpenCV 3.0. En cuanto al software de orquestación, mantenimiento y gestión ha sido rechazado (complicaba más la configuración).

DESARROLLO

Este proyecto ha sido planteado para ser desarrollado en cuatro fases, la primera ha sido instalación y configuración de un solo nodo, la segunda fase ha sido el estudio de los diferentes softwares para su gestión, la tercera fase ha sido la implementación final del clúster, y finalmente, la cuarta parte ha sido la realización de las pruebas.

- **Primera parte:** Para realizar esta parte, se ha usado la versión de Raspbian completa, incluida en NOOBS, ya que esta Raspberry sería usada para desarrollar el código, cuando se completó la instalación se instaló OpenCV 3.0 para ser usado con Python 3.4. La instalación se ha realizado creando un entorno virtual, esto hace que se aíslen las dependencias, por si hubiera que trabajar con otras versiones de OpenCV o de Python. La instalación de Python requiere que se compilen las librerías de OpenCV este paso puede llegar a tomar en torno a una hora, si todo sale bien, haciendo uso de los 4 núcleos de la Raspberry. Al trabajar en un entorno virtual, para poder trabajar con OpenCV y la versión adecuada de Python, hay que cargar el entorno que necesitábamos, para agilizar el uso del sistema se modificó el fichero RC.local para cargar el entorno virtual al iniciar el sistema. No se hicieron más cambios en la configuración del sistema operativo. Cuando estuvo el sistema listo, se realizó un pequeño código que analizaba las imágenes. Se analizó el uso de gitHub, pero se descartó ya que se empleaba más tiempo en usar la herramienta que en el desarrollo del código. Esta parte se dio por concluida cuando se consiguió procesar una lista de imágenes.

- **Segunda parte:** En esta parte no se volvió a abordar la programación de OpenCV y se empezó a estudiar las diferentes formas de gestionar el clúster, fue en esta parte donde se ha invertido más tiempo. El primer software que se estudió fue Docker, ya que al poder meter OpenCV en un contenedor, se agilizaría el despliegue de la aplicación, ya que la instalación es lenta y delicada. Para construir el clúster, y abaratar costes, se hicieron los cables de red, por otro lado, se imprimió una serie de carcasas para tener las Raspberry montadas, también se consiguió un switch de 16 puertos para interconectar las Raspberry. Cuando ya estaba construido el clúster, físicamente, se comenzó con su configuración, para agilizar las cosas, se analizó el uso de Ansible, pero al ser configuraciones únicas, se descartó su uso. El uso de docker también se justifica con el uso de kubernetes, ya que requiere de docker para poder funcionar, tras ver que el uso de estas tecnologías no aporta lo que se les pedía, se descartó su uso.

- **Tercera parte:** Tras descartar el uso de los diferentes programas para gestionar el clúster, se decidió hacerlo de la manera más sencilla posible. Para ello se usaron las imágenes de Raspbian, la mínima para los 5 nodos y una normal para el máster. En este caso en lugar de trabajar con todas las Raspberry en paralelo, primero se configuró el máster, luego la configuración e instalación de un nodo, para evitar futuros problemas como en el punto

anterior se realizaron copias de seguridad de los dos sistemas operativos, la copia del nodo fue usada en el resto de las Raspberry, de este modo, solo se tarda en torno a 40 minutos en tener una Raspberry totalmente preparada para analizar imágenes. En esta parte se retomó la parte de programación, se depuró el programa y se hizo un programa para aprovechar los 4 núcleos de la Raspberry, también se crearon una serie de códigos para automatizar algunas tareas, como el reparto de imágenes.

- **Cuarta parte:** En esta parte se diseñaron las pruebas, se corrigieron los fallos de código y se llevaron a cabo. En las diferentes pruebas que se realizaron, se recolectaron los siguientes datos de manera secuencial: Tiempo en procesar las imágenes, y tiempo en cargar en memoria RAM las imágenes y procesarlas. Y, en el clúster: Mayor tiempo en procesar las imágenes, mayor tiempo en cargar las imágenes y procesarlas, y tiempo en distribuir las imágenes. Y se calculó la aceleración del mayor tiempo en procesar las imágenes en el clúster, que en teoría rondará 6, y la aceleración contando la transferencia de imágenes contra el tiempo de hacerlo en secuencial.

RESULTADOS Y DISCUSIÓN

Para hacer las pruebas se ha tenido en cuenta tanto los tamaños de las memorias caché como el número de Raspberry disponibles, siendo el tamaño de caché 32 KB y de 512 KB. Se ha decidido hacer uso de fotos de tamaños de 16 KB a 1 MB. Con esta distribución, con 16KB y con 32Kb las fotos caben en el primer nivel de caché, con 64 KB y 128 KB, las fotos caben en el nivel L2, finalmente con 256 KB, 512 KB y 1 MB no cabrían en L2 y tendremos que ir a RAM. Posteriormente, para seguir probando el sistema, se añadieron fotos de 2 MB, 4 MB y 8 MB ya que aún quedaba margen de mejora. En cuanto al número de imágenes que se usaron fueron, 1, 6, 12, 24, 48, 96, 192 y 196. El número 196 fue elegido por no ser múltiplo del 6; por tanto, el reparto no será equitativo entre los nodos.

Para comparar la aceleración, primero se analizaron las imágenes en secuencial, y guardé el tiempo que se tarda en procesar las imágenes y el tiempo que se tarda en cargar las imágenes y en procesarlas. Luego, se analizaron las imágenes distribuyéndolas por el clúster, se tomaron los tiempos que necesitaba en procesar las imágenes, el tiempo en cargarlas y procesarlas y el tiempo que tardaba en distribuir las imágenes.

Para realizar las pruebas se ha usado la siguiente imagen [20]:

Figura 2. Imágenes de prueba.

Los resultados de las pruebas se han separado en 3 grupos: secuencial, clúster y aceleración. En la columna secuencial: "proc" es el tiempo en procesar las imágenes y "total" es el tiempo en cargar las imágenes y procesarlas. En la columna de clúster: "proc" es el tiempo de las Raspberry más lenta en procesar las imágenes, "cargas" es el tiempo de las Raspberry más lenta en cargar las imágenes y procesarlas, "trans" es el tiempo en distribuir las imágenes y "total" es la suma de cargas y transmisión de ida y vuelta, el valor de vuelta se ha considerado igual que el de ida. Por último, en la columna de la aceleración: "aceleración" (en verde) es la aceleración de sólo contra procesamiento y "aceleración" (en naranja) es la aceleración de total (clúster) con respecto del total (secuencial). Todos los tiempos están en segundos, indicado entre paréntesis en las etiquetas.

Cuando se estaba analizando que tipo de pruebas se iban a realizar, se pensó que el tiempo de carga de las imágenes podría afectar al rendimiento, por eso se ha tomado ese tiempo. Como veremos en las pruebas, no afecta al rendimiento.

Imágenes de 256KB

	SECUENCIAL		CLÚSTER				ACELERACIÓN	
256kb	proc (s)	total (s)	proc (s)	cargas (s)	trans (s)	total (s)	Parcial	Total
1	2,1035	2,3853	-	-	-	-	-	-
6	12,5861	12,9292	2,2093	2,5002	2,8138	8,127	5,6968	1,5907
12	25,619	24,6087	4,2816	4,5874	5,0882	14,763	5,9835	1,6668
24	50,3359	50,9133	8,4197	8,672	10,2115	29,095	5,9783	1,7498
48	100,9222	101,8289	16,7868	17,1454	20,1181	57,381	6,0119	1,7745
96	201,8187	203,3095	33,5992	34,0461	39,5084	113,069	6,0066	1,7981
192	403,0051	405,6459	67,0557	67,719	81,9585	231,636	6,0100	1,7512
196	411,8935	414,6502	75,5364	76,2097	80,2309	236,671	5,4529	1,7520

Aunque las fotos van aumentando de tamaño, no afectan a los tiempos de carga, comparados con el procesamiento. Y por primera vez conseguimos acelerar el cómputo al usar el clúster:

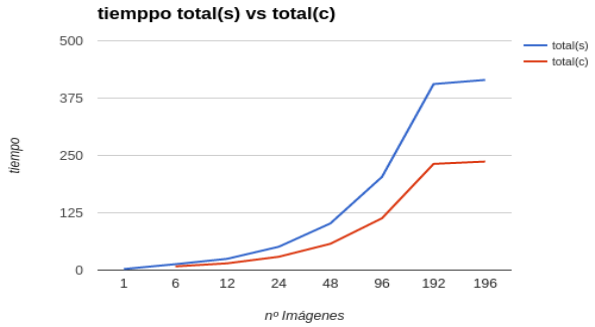

Figura 3. Comparativa de tiempos entre clúster y secuencial para imágenes de 256KB.

Se ha conseguido mejorar el proceso en torno al 60%- 75%.

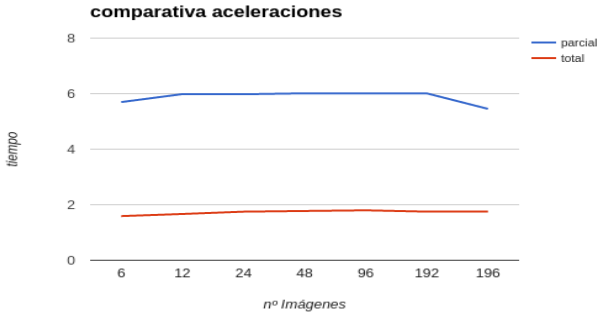

Figura 4. Comparativa de aceleraciones para imágenes de 256KB.

Imágenes de 1MB

	SECUENCIAL		CLÚSTER				ACELERACIÓN	
1Mb	proc (s)	total (s)	proc (s)	cargas (s)	trans (s)	total (s)	Parcial	Total
1	10,213	10,54	-		-	-	-	-
6	61,264	61,84	10,15	15,500	3,447	22,394	6,0358	2,7614
12	122,599	123,48	20,398	20,772	6,2472	33,267	6,010	3,7117
24	244,509	246,509	40,676	41,138	12,112	65,364	6,011	3,7713
48	489,180	492,037	81,765	82,409	23,637	129,68	5,9827	3,7940
96	982,032	987,307	162,686	163,703	48,506	260,71	6,036	3,7869
192	1964,90	1974,805	325,019	326,873	94,637	516,14	6,045	3,8260
196	2005,32	2015,689	366,464	368,418	100,520	569,45	5,4720	3,5396

Imágenes de 2MB

	SECUENCIAL		CLÚSTER				ACELERACIÓN	
2Mb	**proc (s)**	**total (s)**	**proc (s)**	**cargas (s)**	**trans (s)**	**total (s)**	**Parcial**	**Total**
1	24,0345	24,4115	-	-	-	-	-	-
6	144,3368	145,2066	24,1336	24,4998	3,83	32,1598	5,9807	4,5151
12	289,5328	289,9776	48,1788	48,645	7,7606	64,1662	6,0095	4,5191
24	576,0753	578,6906	96,066	96,725	14,9865	126,698	5,9966	4,5674
48	1150,683	1155,6488	191,7229	192,802	32,8888	258,5796	6,0018	4,4692
96	2311,518	2321,758	383,9906	385,8917	68,0996	522,0909	6,0197	4,4470
192			768,1628	771,8428	125,462	1022,767	-	-
196			863,4887	867,6123	127,804	1123,2219	-	-

Las celdas coloreadas en rojo son debido a falta de RAM. Cuando esto pasa, el sistema aborta la ejecución del programa. Esto pasa porque, lo primero que se hace para analizar las imágenes es cargar las imágenes. Por último, en esta gráfica podemos ver todas las aceleraciones juntas:

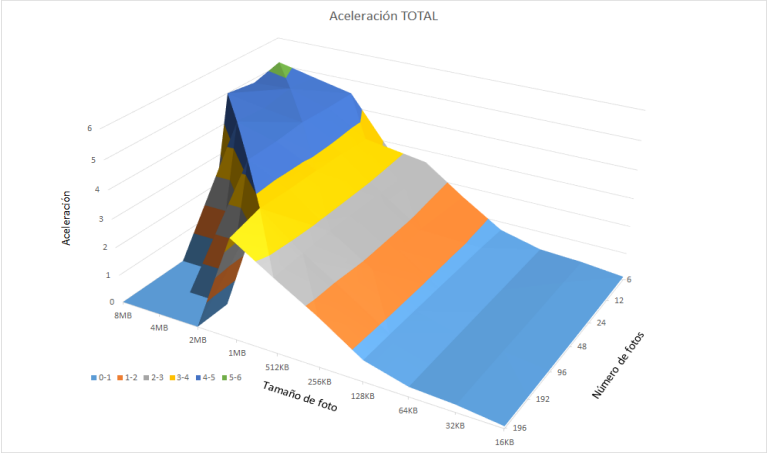

Figura 5. Comparativa global para todos los tamaños de fotos, cantidad de elementos y aceleración resultante.

REFERENCIAS BIBLIOGRÁFICAS

[1] Raddick, J., *et al.* (2007). Galaxy Zoo: an experiment in public science participation. En *Bulletin of the American Astronomical Society*, p. 892.

[2] Patterson, D. A., Hennessy, J. L., y Goldberg, D. (1990). *Computer architecture: a quantitative approach.* San Mateo, CA: Morgan Kaufmann.

[3] Culler, D., Singh, J.P., y Gupta, A. (1999). Parallel computer architecture: a hardware/software approach. *Gulf Professional Publishing.*

[4] Janard, K. y Marurngsith, W. (2015). Accelerating real-time face detection on a raspberry pi telepresence robot. En *Innovative Computing Technology (INTECH), 2015 Fifth International Conference on IEEE,* pp. 136-141.

[5] Cloutier, M.F., Paradis, C., y Waever, V.M. (2016). A raspberry pi cluster instrumented for fine-grained power measurement. *Electronics*, 5(4), p. 61.

[6] Pajankar, A. (2015). *Raspberry Pi computer vision programming*. Packt Publishing Ltd.

[7] Kiepert, J. (2013). *Creating a raspberry pi-based beowulf cluster.* Boise State University, pp. 1-17.