Capitulo 3: Integración de los Sistemas Embebidos

Luego de conocer los diferentes tipos de Manejadores de Paquetes, es necesario conocer la infraestructura que se usaría en el proyecto. Para ello se inició con ciertas tarjetas de programación general, esto con el fin de entender el poder o lo que son capaces de hacer estas tarjetas.

A través del capítulo se especificará las tarjetas que probamos, y nuestros resultados, ventajas o desventajas de ellas. Al final del capítulo se indicará que sistema embebido se usó para hacer las pruebas de rendimiento computacional y energético.

Adapteva Parallella Board

Las tarjetas Parallella de la empresa Adapteva [1] son pequeñas tarjetas embebidas del tamaño de una tarjeta de crédito ideales para la gente que desee empezar en el mundo de la computación paralela. Desde una perspectiva más general, estas tarjetas son buenas para la enseñanza de la computación paralela debido a su fácil configuración y su infraestructura.

Consta de un procesador Zynq-Z7010 Dual core ARM A9 y un co-procesador Epiphany de 16 cores. Tiene 1GB de memoria RAM, slot para tarjeta MicroSD, conexión USB 2.0, ethernet y soporta Linux como sistema operativo (probamos ubuntu). [1]

Al momento de usarla nos dimos cuenta de ciertos aspectos. Los procesadores se sobrecalientan mucho, así que al hacer operaciones muy pesadas, el fabricante indica que requiere un ventilador o algún objeto para mantenerlo frio. En el caso del proyecto, se utilizó dentro de un entorno frío (laboratorio con aire acondicionado) y luego se usó un ventilador al tenerlo fuera del laboratorio.

En forma general se instaló una imagen del sistema operativo en la tarjeta SD y luego se pudo instalar y probar Python y OpenMPI/MPI para uso paralelo con los ejemplos del paquete.

La *Parallella* es un buen sistema integrado que tiene múltiples aplicaciones. Desde el punto de vista del proyecto es ideal para operaciones de alto rendimiento debido a su posibilidad de realizar operaciones paralelas. Ésta gran característica permite que laboratorios pequeños puedan realizar pruebas de sus operaciones o que los colegios puedan acceder al aprendizaje práctico de la computación paralela usando un dispositivo con un pequeño costo.

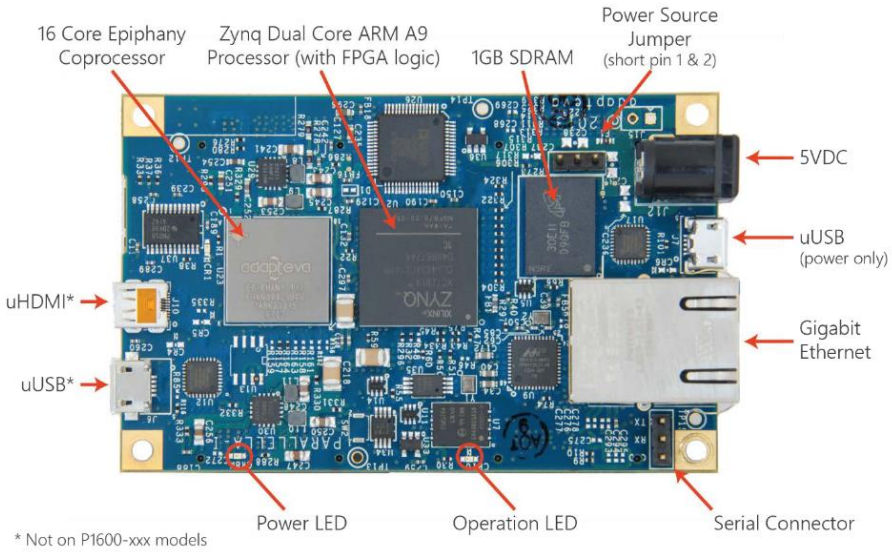

Ilustración 1 Tarjeta Parallella de la empresa Adapteva.

No obstante, y a pesar de que no es una desventaja en sí sino cuando se compara con otros sistemas, a falta de GPU no se puede acelerar mucho los procesos y a pesar de que el consumo energético sea poco con respecto a otros sistemas de procesadores, siguen siendo procesadores que gastan un mayor consumo que un GPU.

Además de lo anterior, si los programas requieren escalarse no va a ser tan sencillo y tendrá una limitante: su memoria. Las tarjetas SD son la única fuente de memoria que tiene la tarjeta, incluso allí es donde se quema la imagen del sistema operativo con todos sus archivos. Por consiguiente, usarlo para procesos de HPC en donde los datos sean muy extensos será una gran limitante para que el sistema funcione. No obstante, cabe volver a resaltar que para otras operaciones computacionales este sistema es capaz de realizarlo.

Intel Galileo Board

De la página official de Arduino, la Galileo es un microcontrolador basado en el procesador Intel Quark SoC X1000 DE 32-bits [2]. Es una tarjeta usada para propósito general que tiene funcionalidades expansivas gracias a que tiene entradas que posee Arduino (por eso es certificada por Arduino). Por lo anterior entonces se conoce que esta tarjeta abarca aplicaciones generales y algoritmos electrónicos (aquellos que se realizan con tarjetas como el Arduino).

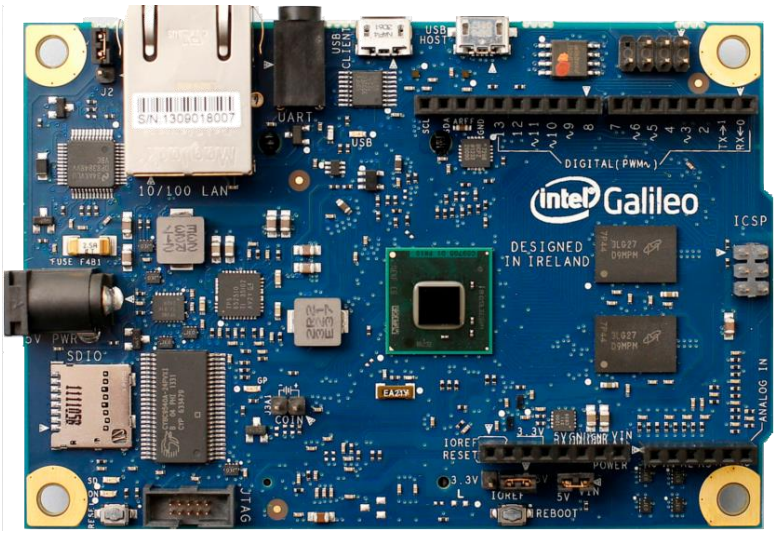

Ilustración 2 Vista superior de la Intel Galileo Gen 1

Como se observa en la Ilustración 2, la Intel se compone de un procesador, conexión ethernet y un sin número de pines de tipo Arduino. Tiene conexiones USB para depuración, funciona con 5V de energía y tiene entrada para memoria externa micro SD hasta de 32 Gb.

Las ventajas de esta tarjeta es otra vez el consumo energético y su extensibilidad de funcionalidad para otro tipo de operaciones. Sin embargo, para *High Performance Computing (HPC)*, esta tarjeta carece de potencia suficiente para realizar estas operaciones; no esta diseñada para este tipo de problemas, por lo que no soporta siquiera operaciones paralelas.

Además de lo anterior, se quiso usar un mini cluster de estos sistemas, para explotar su bajo consumo energético, pero la Generacion 1 de Intel Galileo (la que está en la Ilustración 2), no soporta ser un sistema esclavo, por lo que no existe manera de usarlo en sistemas distribuidos.

Sin embargo, la generación 2 sí se podía usar como un cluster por su soporte a ser sistema esclavo. Existe un proyecto que logró conectarlos y hacer pruebas pero basados en sistemas como Hadoop. -revisar esta bibliografía-

Al iniciar este proyecto, la Galileo Generación 1 ya se encontraba descontinuada, y la Generación 2 se descontinuó a mediados del 2017 [3], dejando sin salida el uso de este sistema.

NVIDIA Jetson TK1

La NVIDIA Jetson TK1 [4] es un sistema embebido de propósito general cuyo principal objetivo es explotar las capacidades de cómputo de sus 192 CUDA Cores que posee. Este sistema embebido tiene la idea de poder usar algoritmos paralelos y de alto rendimiento, explotando los GPU y consumiendo menos energía.

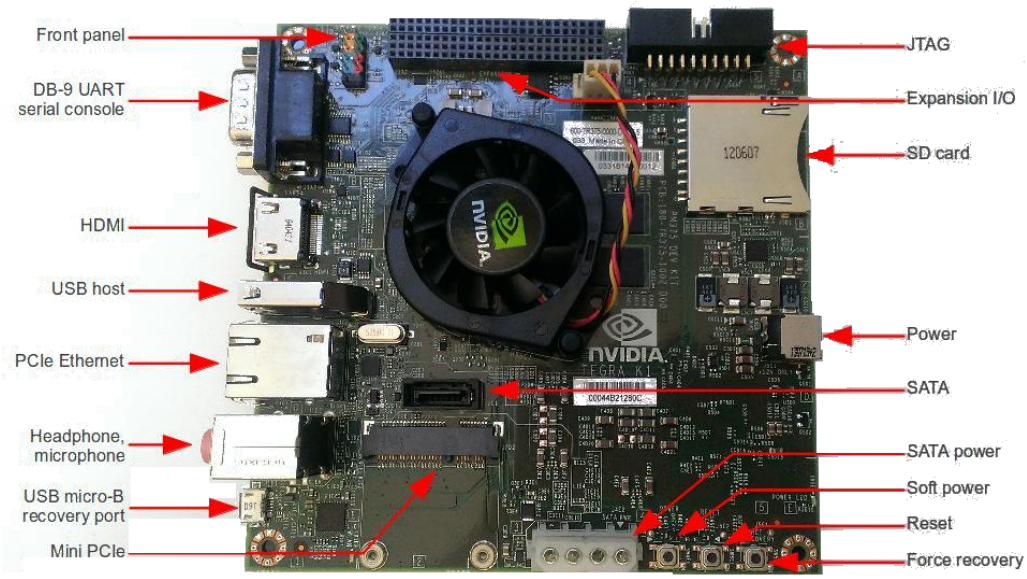

Ilustración 3 NVIDIA Jetson TK1

La Jetson TK1 consta de una Tegra K1 SoC (de ahí el nombre) con un procesador NVIDIA ARM Cortex-A15 Quad-core y NVIDIA Kepler GPU con 192 CUDA Cores. Posee una memoria eMMC de 16GB, una memoria RAM de 2GB, puerto HDMI, conexión ethernet, slot para tarjeta SD, puerto USB para teclado y ratón (uno solo) y entrada serial para otro tipo de conexiones (cuando falla la USB, por ejemplo).

Las nuevas características de esta arquitectura son la *Hyper-Q* que hace que el GPU trate de estar ocupado siempre que pueda y *Dynamic Paralelism* que permite que el GPU cree nuevas tareas, aliviando al procesador de crearlas.

En el proyecto, se quemó el sistema operativo usando Nvidia Linux 4Tegra (L4T) Board Support Package (BSP), y Sample Root Filesystem from Nvidia. Este sistema viene con Ubuntu pero tiene todas las librerias y programas para realizar algoritmos paralelos, distribuidos y el uso de las GPU con CUDA.

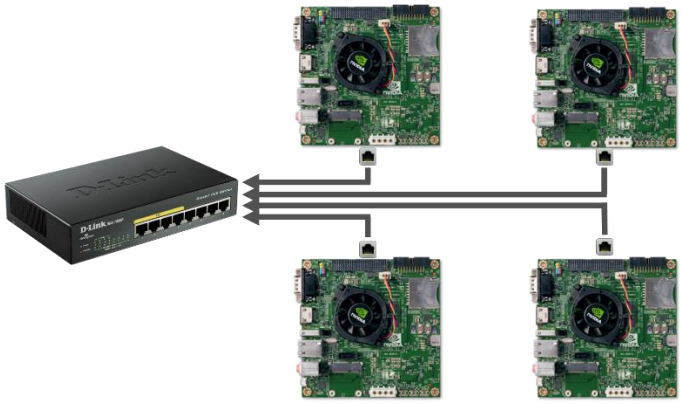

Ilustración 4 Infraestructura de los 4 Jetson TK1 hechos como prototipo de pruebas

Con respecto a su infraestructura, tal como lo muestra la Ilustración 4, se usaron 4 Nvidia Jetson TK1 conectados por cable ethernet a un pequeño router (distinto a la de la imagen). Cada uno de ellos se le instaló el L4T, y, aunque se puede instalar fácilmente mediante la herramienta Jetpack de NVIDIA, se requiere usar solamente la distribución Ubuntu para poder usarla, así que se requirió hacerlo de otro modo.

Cada TK1 se revisó que funcionara las librerias de CUDA y de MPI. Para ello, luego de instalar CUDA (que a propósito también tuvo problemas en su instalación al no tener la Jetpack) se usaron los algoritmos de prueba que vienen con la distribución para revisar su funcionamiento, y luego de instalar MPI se siguió un tutorial para configurar correctamente el sistema y probarlo con el mismo algoritmo de prueba que en ella se encuentra [5].

El proceso de su instalación y acople fue exitoso. No obstante, tardó demasiado tiempo acostumbrarse y encontrar soluciones a los problemas, y todo esto debido a su madurez del sistema (es el primero de este tipo de NVIDIA, por lo que los errores y compatibilidades se estaban apenas analizando). Luego del éxito usando librerias tradicionales se pasó a empaquetarlas en Nix y lastimosamente la versión de Nix para ARM no era compatible para este sistema embebido y sin forma alguna de instalarlo se decidió probarlo con NixOS, teniendo un resultado peor ya que requería el uso de un conector Serial para poder comunicarse con el sistema, debido a que ningún otro puerto era compatible (y habría que volverlos compatible a mano, decidiendo que esta tarea estaba lejos de los objetivos del proyecto).

Estos sistemas son fácilmente escalables, realizan paralelismo y además explotan las capacidades de la GPU. También efectúan operaciones con bajo consumo de energía por lo que era una buena candidata para el análisis de un sistema para HPC usando manejadores de paquetes.

La desventaja de este sistema fue su tiempo. Al día de hoy la tarjeta ya dejó de producirse, por lo tanto ya se encuentra obsoleta. Esto repercute en las aplicaciones y librerias que ya no se actualizan con soporte a este tipo de sistemas sino a las versiones posteriores. Esto se vio reflejado en el uso de NIX dentro de este sistema, que al inicio del levantamiento de la infraestructura no era compatible por estar en versión beta el soporte a sistemas embebidos, pero más adelante el soporte se usó en Tegras TX1 y TX2. En los capítulos siguientes se ampliará con detalle este proceso y sus alti-bajos.

NVIDIA Jetson TX1

La NVIDIA Jetson TX1 (o TX1 para abreviar) es un sistema embebido desarrollado por NVIDIA que es la versión siguiente a la Jetson TK1. Posee la arquitectura Maxwell con 256 *CUDA Cores*, un procesador Quad ARM A57 con 2MB de cache, 16GB de memoria eMMC, entrada SDIO y SATA, conexión USB 3.0 y 2.0, conectividad LAN, WLAN y bluetooth, además de que reproduce videos hasta 4K.

Este sistema embebido mejora su versión de la TK1 de gran manera gracias a su nuevo procesador y la nueva cantidad de GPUs. Está diseñado para ejecutar eficientemente programas de *Deep Learning*, visión por computadora, *GPU Computing* y gráficas, siendo ideal para Inteligencia Artificial [6]. De hecho, la publicidad sobre este sistema ha sido su gran capacidad de cálculo en AI (*Artificial Intelligence*).

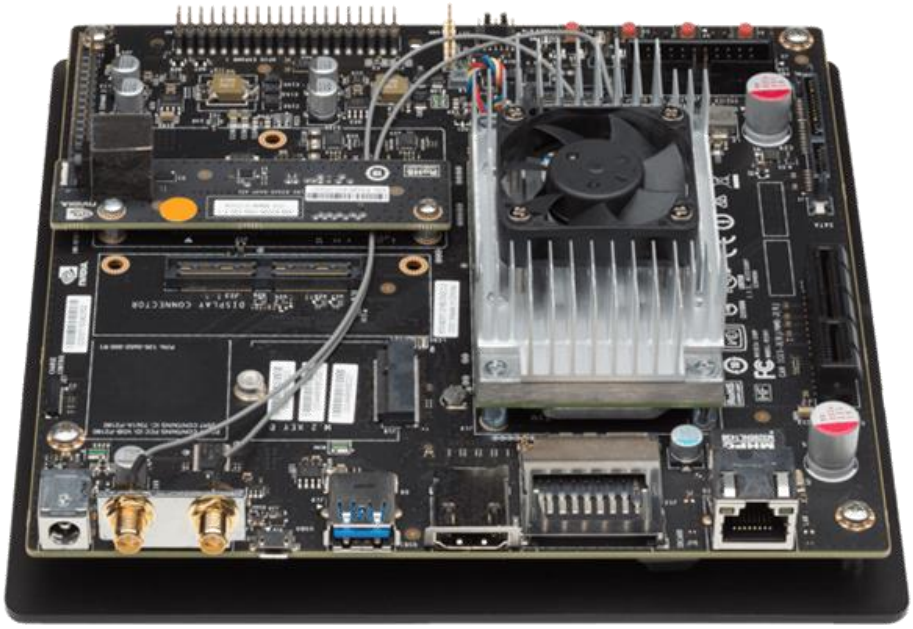

Ilustración 5 NVIDIA Jetson TX1

Como la Jetson TX1 es el sistema embebido que tiene mejores especificaciones que cualquier otro sistema probado, se decide hacer con este sistema las respectivas pruebas de rendimiento.

Se usaron 8 NVIDIA Tegra TX1 interconectadas con un *switch* y dejando un portátil como sistema maestro. Cada uno de ellos se instaló las librerias CUDA y MPI, probándolas por separado y luego en conjunto como un sistema distribuido. Cada prueba se basó en los ejemplos que vienen por defecto en las instalaciones de CUDA y de un tutorial de MPI que incluso mostraba como configurar el sistema [5].

En el siguiente capítulo se detallará más el proceso que se realizó de la conexión y pruebas de la Jetson TX1, además de su implementación con el Nix, el manejador de paquetes funcional.

-TABLA DE LAS VENTAJAS Y DESVENTAJAS DE LAS TEGRAS (Y LA OTRAS IGUAL INDICANDO EL PORQUE NO)-

[1] «The Parallella Board,» [En línea]. Available: https://www.parallella.org/. [Último acceso: Marzo 2019].

[2] «Intel Galileo (Arduino web page),» [En línea]. Available: https://www.arduino.cc/en/ArduinoCertified/IntelGalileo. [Último acceso: Marzo 2019].

[3] «Intel Galileo Gen 2 Board (Specs),» [En línea]. Available: https://ark.intel.com/content/www/us/en/ark/products/83137/intel-galileo-gen-2-board.html. [Último acceso: Marzo 2019].

[4] «Jetson TK1,» [En línea]. Available: https://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html. [Último acceso: Marzo 2019].

[5] «MPI Tutorial,» [En línea]. Available: http://mpitutorial.com/. [Último acceso: Marzo 2019].

[6] «Nvidia Jetson Systems,» [En línea]. Available: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems-dev-kits-modules/. [Último acceso: Marzo 2019].

[7] «Docker Get Started,» [En línea]. Available: https://docs.docker.com/get-started/. [Último acceso: Febrero 2019].

[8] «Docker Web Page,» [En línea]. Available: https://www.docker.com/what-container#/virtual_machines.

[9] «What is Heterogeneous Computing?,» 2017. [En línea]. Available: http://developer.amd.com/resources/heterogeneous-computing/what-is-heterogeneous-computing/.

[10] «What is edge computing and how it's changing the network,» [En línea]. Available: https://www.networkworld.com/article/3224893/internet-of-things/what-is-edge-computing-and-how-it-s-changing-the-network.html.

[11] F. Bonomi, R. Milito, J. Zhu y S. Addepalli, «Fog Computing and Its Role in the Internet of Things,» ACM, 2012.

[12] S. Yi , Z. Hao, Z. Qin y Q. Li, «Fog Computing: Platform and Applications,» IEEE, 2015.

[13] Y. Ai, M. Peng y K. Zhang, «Edge Computing Technologies for Internet of Things: A Primer,» Elsevier, 2017.

[14] S. Yi, C. Li y Q. Li, «A Survey of Fog Computing: Concepts, Applications and Issues,» 2015.

[15] P. Lucas, J. Ballay y M. McManus, Trillions: Thriving in the Emerging Information Ecology, 2012.

[16] «Package Manager,» [En línea]. Available: https://en.wikipedia.org/wiki/Package_manager. [Último acceso: 2017].

[17] «Nix Package Manager,» [En línea]. Available: https://nixos.org/nix/about.html.

[18] «NixOs,» [En línea]. Available: https://nixos.org/.

[19] M. Geveler, D. Ribbrock, D. Donner, H. Ruelmann, C. Hoppke, D. Schneider, D. Tomaschewski y S. Turek, «The ICARUS White Paper: A Scalable Energy-Efficient, Solar-Powered HPC Center Based on Low Power GPUs,» Springer, 2017.

[20] «Montblanc-Project,» [En línea]. Available: http://www.montblanc-project.eu/.

[21] J. Saffran, G. Garcia, M. Souza, P. Penna, M. Castro, L. Góes y H. Freitas, «A Low-Cost Energy-Efficient Raspberry Pi Cluster for Data Mining Algortihms,» Springer, 2017.

[22] F. P. Tso, D. White, S. Jouet, J. Singer y D. Pezaros, «The Glasgow Raspberry Pi Cloud: A Scale Model for Cloud Computing Infraestructures».

[23] B. Bzeznik, O. Henriot, V. Reis, O. Richard y L. Tavard, «Nix as HPC Package Management System,» ACM, 2017.

[24] M. Plauth y A. Polze, «Are Low-Power SoCs Feasible for Heterogeneous HPC Workloads?,» Springer, 2017.

[25] J. Guerreiro, A. Illic, N. Roma y P. Tomás, «Performance and Power-Aware Classification for Frequency Scaling of GPGPU Applications,» Springer, 2017.

[26] N. Saurabh, D. Kimovski, S. Ostermann y R. Prodan, «VM Image Repository and Distribution Models for Federated Clouds: State of the Art, Possible Directions and Open Issues,» Springer, 2017.

[27] J. Breitbart, S. Pickartz, J. Weidendorfer y A. Monti, «Viability of Virtual Machines in HPC,» Springer, 2017.

[28] «Amazon Web Services,» 2017. [En línea]. Available: https://aws.amazon.com/es/.

[29] «RISC vs CISC,» 2000. [En línea]. Available: https://cs.stanford.edu/people/eroberts/courses/soco/projects/risc/risccisc/index.html. [Último acceso: 2017].

[30] T. Gamblin, M. LeGendre, M. R. Collette, G. L. Lee, A. Moody, B. R. de Supinski y S. Futral, «The Spack Package Manager: Bringing Order to HPC Software Chaos,» ACM, 2015.

[31] «Easy Build Web Page,» [En línea]. Available: https://easybuild.readthedocs.io/en/latest/. [Último acceso: Febrero 2019].

[32] N. Rajovic, A. Rico, N. Puzovic y C. Adeniyi-Jones, «Tibidabo: Making the case for an ARM-based HPC system,» Elsevier, 2013.

[33] B. Bashari Rad, H. John Bhatti y M. Ahmadi, «An Introduction to Docker and Analysis of its Performance,» IJCSNS International Journal of Computer Science and Network Security, 2017.

[34] M. Geimer, K. Hoste y R. McLay, «Modern Scientific Software Management Using EasyBuild and Lmod,» First International Workshop on HPC User Support Tools, New Orleans, 2014.

[35] «Spack: A flexible package manager that supports multiple versions, configurations, platforms, and compilers.,» [En línea]. Available: https://spack.io/. [Último acceso: Febrero 2019].

[36] E. Dolstra, The Purely Functional Software Deployment Model, Utrecht, 2006.