ECMRE: Extended Concurrent Multi Robot Environment

J. Castro, L. De Giusti[1,3], G. Gorga[1], M. Sánchez[1], M. Naiouf[1], A. De Giusti[1,2]

[1] Instituto de Investigación en Informática LIDI (III-LIDI) – Facultad de Informática –UNLP
[2] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
[3]Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC)
camaju_25@hotmail.com,{ldgiusti,ggorga,msanchez,mnaiouf,
degiusti}@lidi.info.unlp.edu.ar

Abstract. El entorno ECMRE es una extensión de CMRE (Concurrent Multi Robot Environment) donde se incorporan características relacionadas a las arquitecturas paralelas actuales: heterogeneidad de procesadores, consumo energético, técnicas de cambios de velocidad del procesador en relación a la temperatura y/o consumo energético.
ECMRE permite incorporar los temas de concurrencia y paralelismo de forma sencilla y amena en cursos iniciales de las carreras de Informática mediante un entorno gráfico e interactivo.
Se ha realizado una prueba inicial en un curso con 42 alumnos para analizar la adaptación a este nuevo entorno y la utilización que pueden hacer del mismo.

***Keywords**:* Concurrencia, Paralelismo, Procesadores heterogéneos, Algoritmos paralelos, Consumo energético.

1 Introducción

La Concurrencia ha sido un tema central en el desarrollo de la Informática y los mecanismos de expresión de procesos concurrentes que cooperan y compiten por recursos ha estado en el núcleo curricular de los estudios de Informática desde la década del 70, en particular a partir de los trabajos fundacionales de Hoare, Dijkstra y Hansen [1][2][3]. Por su parte el paralelismo, entendido como "concurrencia real" en la que múltiples procesadores pueden operar simultáneamente sobre varios threads o hilos de control en el mismo instante, resultó durante muchos años una posibilidad limitada por la tecnología de hardware disponible [4]. Las currículas informáticas clásicas [5][6][7] contenían conceptos de concurrencia en diferentes áreas (Lenguajes, Paradigmas, Sistemas Operativos) pero omitían casi totalmente el tratamiento del paralelismo, salvo al plantear los conceptos de sistemas distribuidos.

Los cambios tecnológicos han producido una evolución de los temas de mayor interés en informática, fundamentalmente por las nuevas aplicaciones que se desarrollan a partir de disponer de arquitecturas y redes de comunicación de mayor potencia y menor costo [8]. Por este motivo, las recomendaciones curriculares internacionales mencionan la necesidad de tratar los temas de concurrencia y paralelismo desde las etapas tempranas de la formación del alumno, dado que todas las arquitecturas y sistemas reales con los que trabajará son esencialmente paralelos

[9]. Sin embargo, la programación paralela (y los conceptos fundamentales de concurrencia) resulta más compleja para un alumno en las etapas iniciales de su formación, y es necesario contar con nuevas estrategias que permitan abordarlos.

Dados los estímulos que los alumnos reciben desde temprana edad, ya sea mediante juegos electrónicos, computadoras, celulares, tablets o cualquier otro dispositivo electrónico, la utilización de herramientas interactivas para la enseñanza de conceptos básicos a alumnos desde un curso CS1 [9][10][11] se ha vuelto una herramienta fundamental [12]. En este sentido, la posibilidad dar los primeros pasos en el mundo de la programación mediante un entorno gráfico e interactivo permite reducir la brecha que tradicionalmente existió entre la abstracción y la posibilidad de ver gráficamente la aplicación de los conceptos estudiados en un entorno que conceptualmente es similar a los utilizados en la vida cotidiana [8][13].

El entorno gráfico CMRE, donde se cuenta con un conjunto de robots que se mueven en una ciudad, ha permitido incorporar la enseñanza de los conceptos básicos de concurrencia y paralelismo en un curso inicial de Informática. En un artículo previo [14] se ha presentado la idea de incorporar al entorno características avanzadas de las arquitecturas paralelas actuales (tales como heterogeneidad, consumo de energía, temperatura generada). Continuando con ese trabajo, se ha realizado dicha extensión generando el entorno ECMRE (Extended Concurrent Multi Robot Environment), el cual se presenta en este trabajo.

El artículo está estructurado de la siguiente manera: en la Sección 2 se detallan las características avanzadas de las arquitecturas paralelas modernas que se han incluido en el entorno; en la Sección 3 se presenta el entorno CMRE actual. En la Sección 4 se explican las extensiones realizadas para generar el ECMRE; mientras que en la Sección 5 se muestra una prueba del nuevo entorno en un curso de primer año. En la Sección 6 se detallan las conclusiones.

2 Características avanzadas de Arquitecturas Paralelas

Las arquitecturas paralelas actuales poseen características avanzadas que son importantes de incluir dentro de los conocimientos iniciales de concurrencia y paralelismo. En particular la heterogeneidad de la arquitectura y el consumo energético.

2.1 Heterogeneidad de las Arquitecturas Paralelas

Históricamente se ha buscado incrementar el poder computacional de las máquinas. Sin embargo, se ha llegado a una situación en la que resulta difícil acelerar la velocidad de los procesadores incrementando la frecuencia de reloj de los mismos. Son dos los problemas que los arquitectos de hardware deben enfrentar: la generación de calor y el consumo de energía. La solución que presentan los diseñadores a estos problemas ha sido integrar, dos o más, núcleos computacionales dentro de un mismo chip, lo cual se conoce como procesador multicore o multinúcleo. Los procesadores multicore mejoran el rendimiento de una aplicación al distribuir el trabajo entre los núcleos disponibles [15][16].

Actualmente la investigación se está enfocando principalmente en Arquitecturas Multicores Heterogéneas (poseen *cores* con diferentes características en cuanto a rendimiento y consumo de energía, pudiendo utilizar o no distintos set de instrucciones), dado que tener *cores* de diferentes tipos permite optimizar el rendimiento, y al realizar una correcta distribución de las tareas entre los núcleos, se logra una mayor eficiencia en la relación rendimiento/energía.

En este tipo de arquitectura, la heterogeneidad se da en diferentes aspectos, los más importantes son: la potencia de cómputo (velocidad de cómputo) de los *cores*; el tiempo de acceso a memoria y la velocidad de comunicación entre *cores*. Estos tres aspectos definen el tiempo de ejecución de las instrucciones en cada *core*, por lo que una misma sentencia ejecutada en dos núcleos diferentes puede insumir tiempos distintos. Por otro lado, al existir un cierto grado de independencia entre las características que producen la heterogeneidad, no todas las instrucciones se ven influenciadas en la misma proporción. Es decir, una operación de punto flotante ejecutada en el *core* A, puede tardar la cuarta parte del tiempo que en el *core* B, mientras que en el caso de una operación de escritura puede tardar la mitad al ejecutarse.

2.2 Consumo Energético de las Aplicaciones Paralelas

El consumo energético es un punto clave en los procesadores actuales. En general la performance de un algoritmo paralelo no se mide sólo por el tiempo de ejecución del mismo, sino también por la energía consumida. Así aparecen índices que relacionan Flops/Watt o Flops/Joule según se relacione el cómputo con potencia instantánea o energía total [17][18].

Resulta importante en la formación de los alumnos de Informática hacer hincapié en las métricas de consumo como un indicador de calidad de los algoritmos. Asimismo, comprender los mecanismos automáticos que desarrollan los procesadores en función de la temperatura que alcanzan (que es una función directa de la energía consumida en un intervalo de tiempo) [17].

Existen técnicas de ajuste de rendimiento que utilizan los procesadores actuales tomando el consumo energético, la temperatura, y otros valores como indicadores para toma de decisiones. *Overclocking* y *underclocking* son dos de las técnicas más utilizadas para aumentar o disminuir la frecuencia del *clock* del procesador con el objetivo de incrementar la performance o normalizar los valores de consumo y temperatura cuando el procesador se encuentra sobrecargado.

3 El entorno CMRE actual

Las características principales del entorno CMRE pueden resumirse de la siguiente manera [15] [19]:

- Existen múltiples procesadores (robots) que realizan tareas y que pueden cooperar y/o competir. Los mismos representan los "*core*s" de una arquitectura multiprocesador real. Estos robots virtuales pueden tener un reloj propio y diferentes tiempos para la ejecución de sus tareas específicas.

- El modelo de ambiente ("ciudad") en la que desarrollan sus tareas admite áreas privadas, parcialmente compartidas y totalmente compartidas. En un área privada sólo puede moverse un único robot, en un área parcialmente compartida se especifica el conjunto de robots que pueden moverse en ella y en un área totalmente compartida todos los robots definidos en el programa pueden moverse dentro de ella.
- Si se instancia a un sólo robot en un área que abarque toda la ciudad, se repite el esquema del Visual Da Vinci [20][21].
- Cuando dos o más robots están en un área compartida (parcial o totalmente), compiten por el acceso a las esquinas del recorrido y a los recursos que allí existan. Para esto deben sincronizar.
- Cuando dos o más robots (en un área común o no) desean intercambiar información (datos o control) deben hacerlo por mensajes explícitos.
- La sincronización se da por un mecanismo equivalente a un semáforo binario.
- La exclusión mutua puede generarse con la declaración de las áreas alcanzadas por cada robot. Acceder a otras áreas de la ciudad, así como salir de ellas no está permitido.
- Todo el modelo de ejecución es sincrónico y permite la existencia de un reloj virtual de ciclos, que a su vez permite asignar tiempos específicos a las operaciones, simulando la existencia de una arquitectura heterogénea.
- El entorno permite ejecutar el programa de manera tradicional, o paso a paso por instrucciones, dando al usuario un control detallado sobre la ejecución del programa, de manera de poder controlar situaciones típicas de concurrencia tales como conflictos (colisiones) o *deadlocks*.
- En la ejecución paso a paso, el efecto de las operaciones se puede reflejar en los robots físicos, comunicados vía *wifi*. Los robots físicos poseen un sistema operativo Linux que permite ejecutar un servidor http implementado en NodeJS [22]. De esta manera el entorno se comunica con los robots (cada robot físico se corresponderá con uno virtual en el ambiente). La comunicación entre ellos es punto a punto, y bidireccional, es decir, el entorno envía las instrucciones al robot físico y luego este último envía su respuesta al entorno indicando la finalización de la instrucción indicada.

4 Extensiones de CMRE

El entorno ECMRE (Extended Concurrent Multi Robot Environment) incorpora a su versión anterior (CMRE) los conceptos de arquitecturas multicore heterogéneas, el consumo energético, la temperatura de los procesadores y las técnicas de Overclocking y underclocking.

El alumno puede trabajar en ECMRE representando diferentes tipos de arquitecturas multicore y observar en forma gráfica e interactiva información relacionada con los tiempos de trabajo de cada robot y las variaciones de los valores de consumo y temperatura durante la ejecución de su algoritmo. A través del análisis de esta información, el alumno podrá modificar su algoritmo (por ejemplo,

balanceando la carga de trabajo) para obtener soluciones que resulten eficientes desde el punto de vista del consumo de energía y de la temperatura alcanzada.

4.1 Representación de la Heterogeneidad de las Arquitecturas Paralelas

En esta sección se describen las principales adaptaciones efectuadas al entorno CMRE para abordar los aspectos de velocidad y consumo energético incorporados al nuevo entorno ECMRE.

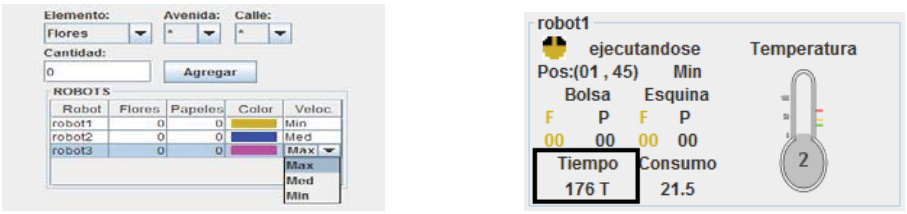
4.1.1 Rendimiento de los procesadores.

En ECMRE existe un área de *Detalles* (Fig.1(a)) donde se configuran características generales de la aplicación, y en particular de cada robot. En ella se incorpora en la tabla ROBOTS, una columna que permite definir la velocidad de cada uno. De esta forma se facilita el trabajo con los robots que podrán mostrar una variación en su rendimiento dado que ejecutarán las instrucciones del algoritmo a distinta velocidad. Existen 3 velocidades elegibles y la relación entre ellas es la siguiente:

- *Max* es la máxima velocidad disponible.
- *Med* es la mitad de la velocidad establecida como velocidad máxima.
- *Min* es la mitad de la velocidad establecida como velocidad media.

En ECMRE se define T como unidad de tiempo (equivale a 100 milisegundos) para medir el rendimiento de los robots/procesadores. Se ha decidido trabajar con un subconjunto de 10 instrucciones primitivas (*bloquearEsquina, depositarFlor, depositarPapel, derecha, enviarMensaje, liberarEsquina, mover, recibirMensaje, tomarFlor* y *tomarPapel*) clasificadas según su grado de complejidad y sobre ellas se asignó un tiempo de ejecución, consecuente con la velocidad del robot.

Además, en ECMRE existe un área de *Información de Ejecución* donde se visualiza la información actualizada de cada robot durante la ejecución del algoritmo (Fig.1(b)), uno de los datos corresponde al tiempo de ejecución en T unidades. Esta información es de gran utilidad para evaluar la performance del algoritmo.



(a) (b)

Fig. 1. Áreas de ECMRE: (a) *Detalles*, donde se configura cada robot, (b) *Información de Ejecución*, donde se observa el tiempo de ejecución del robot.

El manejo de velocidad por robot y la asignación de tiempos a las instrucciones primitivas en ECMRE, permite trabajar con robots/procesadores con distinto rendimiento, simulando una de las características de una arquitectura multicore heterogénea.

4.1.2 Consumo energético

El entorno ECMRE incorpora una nueva sección llamada Robot/Procesador donde se puede parametrizar el consumo energético (en Joules) para un conjunto de instrucciones primitivas (Fig.2(a)). A partir de esta información cada robot almacena su consumo y el mismo es actualizado durante la ejecución del algoritmo, lo cual se visualiza en el área de Información de Ejecución (Fig.2(b)).

(a) (b)

Fig. 2. ECMRE (a) Consumo energético por instrucción en sección Robot/Procesador, (b) *Información de Ejecución*, donde se observa el consumo energético del robot

La velocidad del procesador es un factor que incide sobre el consumo (a mayor velocidad, mayor es el consumo). En ECMRE, el consumo generado por un robot *r* para ejecutar una instrucción *i* está dado por el consumo de *i* especificado previamente multiplicado por un coeficiente que representa a la velocidad de *r* en ese momento (0.25 para *Min*, 0.5 para *Med* y 1 para *Max*).

4.2 Representación de la Temperatura.

En la sección Robot/Procesador mencionada anteriormente también se puede parametrizar, para cada robot, la temperatura producida al ejecutar cada instrucción del conjunto de primitivas (Fig.3(a)), expresado en grados centígrados.

Se ha implementado la lógica necesaria para que cada robot registre su temperatura, que la misma sea actualizada y a la vez pueda visualizarse durante la ejecución del algoritmo. Para ello, se incorporó un termómetro en el área de *Información de Ejecución* del robot (Fig.3(b)) que refleja la temperatura con un valor numérico. El termómetro cambia de color (gris, verde, naranja y rojo) a medida que disminuye o aumenta la temperatura, representando con el color gris a la temperatura mínima y con rojo a la máxima.

(a) (b)

Fig. 3. ECMRE (a) Temperatura por instrucción en sección Robot/Procesador, (b) *Información de Ejecución*, donde se observa la temperatura del robot

Esta característica también está influenciada por la velocidad del robot (a mayor velocidad, mayor es el incremento de la temperatura). En ECMRE, la temperatura de un robot *r* después de ejecutar una instrucción *i* está dado por: la temperatura previa

más la que generada la instrucción *i* especificada previamente, y este total se multiplica por un coeficiente que representa a la velocidad de *r* en ese momento (0.92 para *Min*, 0.95 para *Med* y 0.98 para *Max*).

4.3 Representación de las Técnicas de Overclocking y Underclocking.

Por otra parte, ECMRE contempla la posibilidad de que los robots apliquen *overclocking* y *underclocking* durante la ejecución del algoritmo. En la sección *Robot / Procesador* de ECMRE, se configuran los siguientes parámetros que activan dicha funcionalidad (Fig.4):

- *TCase Max*: es la temperatura máxima del procesador. Si se sobrepasa este valor:
  - Se intentará aplicar *underclock* para disminuir la temperatura del procesador.
  - Si no se admite *underclock* o la velocidad es la mínima, el robot se detiene hasta normalizar su temperatura.
- *Utilizar Overclock*: habilita la utilización de *overclocking* en el robot.
- *Utilizar Underclock*: habilita la utilización de *underclocking* en el robot.
- *Intervalo de Verificación (IV)*: indica cada cuanto tiempo verificar el incremento del consumo energético y aplicar *overclocking/underclocking* según corresponda (en unidades de tiempo T).
- *Variación de Consumo Esperada (VCE)*: indica la variación de consumo energético (en joules) esperada para el tiempo definido en *IV*. Valor utilizado para aplicar *overclocking/underclocking*.

Fig. 4. ECMRE - Parámetros de ajuste de rendimiento en sección Robot / Procesador

El algoritmo de ajuste de rendimiento en cada robot trabaja de la siguiente manera: Por cada instrucción ejecutada evalúa:

- Si la temperatura del robot es mayor a la temperatura máxima especificada (TCase Max):
  - Si el robot soporta underclocking y la velocidad actual no es la mínima, se aplica underclock para disminuir la temperatura. Notar que si la velocidad es Min no es posible seguir disminuyendo su valor.
  - En caso contrario, el robot se detiene hasta enfriar su temperatura a 25 grados y luego continúa su procesamiento.
- Si el incremento de consumo registrado durante el intervalo de verificación (IV), supera la variación de consumo esperada (VCE):
  - Si el robot soporta underclocking y la velocidad actual no es la mínima, se aplica underclock para disminuir el consumo energético y la temperatura. Notar que si la velocidad es Min no es posible seguir disminuyendo su valor.
- Si el incremento de consumo registrado durante el intervalo de verificación (IV), es menor a la variación de consumo esperada (VCE):

- Si el robot soporta overclocking y la velocidad actual no es la máxima, se aplica overclock para incrementar la performance. Notar que si la velocidad es Max no es posible seguir aumentando la velocidad.

Las operaciones de *overclock* y *underclock* no se aplican mientras el tiempo de variación de consumo sea inferior al parámetro *IV*. Cada vez que se aplica *overclock*, *underclock* o el robot se detiene para enfriarse, los valores de variación de consumo y tiempo transcurrido son reiniciados a 0.

4.4 Logs del Procesador.

Las nuevas funcionalidades incorporadas a ECMRE pueden requerir realizar un análisis post ejecución del algoritmo para evaluar si el resultado obtenido cumple con lo esperado. En el caso que, el tiempo de ejecución o el consumo energético sea mayor al esperado, es posible que se proponga modificar la solución implementada o bien reasignar las tareas entre los procesadores involucrados para obtener un mejor resultado. Con el objetivo de facilitar este análisis se ha incorporado a ECMRE una nueva sección llamada *Log del Procesador* que muestra de forma amigable y resumida (mediante la utilización de tablas y gráficos) la ejecución de un algoritmo.

ECMRE registra información de los procesadores durante la ejecución del algoritmo. Dentro de la sección *Robot/Procesador* se puede parametrizar la *frecuencia de log* durante la ejecución de cada robot (cantidad de instrucciones).

Los eventos que se registran pueden ser: procesador detenido por sobrecarga, overclock y underclock. Por cada entrada de log registrada se almacena la siguiente información del robot: evento ocurrido (LOG, DETENIDO, OVERCLOCK y UNDERCLOCK), velocidad antes y después de la ocurrencia del evento, temperatura y consumo energético.

La sección *Log del Procesador* cuenta con 5 subsecciones diferentes denominadas: Temperatura, Gráfico Temperatura, Consumo, Gráfico Consumo y Gráfico Consumo x Instrucción y en cada una de ellas el usuario puede seleccionar el robot objeto de análisis.

5 Sesión de pruebas con alumnos

El entorno ECMRE fue presentado en la materia Taller de Programación de la Facultad de Informática de la UNLP. La materia corresponde al 1er año de las carreras de la Facultad y se compone de 3 módulos, el tercero corresponde a la introducción de los conceptos básicos de la Programación Concurrente, donde actualmente utilizan el entorno CMRE.

La sesión se inició con un breve repaso de los conceptos de concurrencia y paralelismo abordados en el entorno como son las arquitecturas multicore (homogéneas y heterogéneas), consumo energético, temperatura y técnicas de ajuste de rendimiento en procesadores, y balance de carga. Para cada ítem se revisó el concepto y su importancia en relación a las arquitecturas de computadoras actuales y se presentaron los nuevos elementos (temperatura, consumo y velocidad) que han sido incorporados al entorno CMRE dando origen al entorno ECMRE.

Una vez finalizada la revisión teórica, se planteó una actividad práctica que fue desarrollada con la supervisión del docente y el conjunto de los alumnos. Los alumnos no interactuaron directamente con el entorno debido a que aún estaba en la etapa de desarrollo. La actividad práctica consistió en la resolución de un problema modelo, utilizando el entorno ECMRE y donde se mostró la variación en tiempo de ejecución, consumo energético y temperatura de acuerdo a las diferentes configuraciones de los robots, utilizando tablas, gráficos y logs explicados en la sección anterior.

Al finalizar la actividad práctica, se entregó a cada alumno una breve encuesta anónima, con el objetivo de tener un primer feedback de los alumnos, quienes serán usuarios finales de la herramienta. La misma se compone de 5 preguntas cuyas respuestas cubren una escala de Likert que varía de 1 (en completo desacuerdo) a 5 (completamente de acuerdo). La encuesta se realizó a los 42 alumnos que asistieron a la clase de Taller el día de la experiencia y se obtuvieron los resultados de la Tabla 1.

Tabla 1. Resultados encuesta alumnos de Taller de Programación.

PREGUNTA	RESULTADOS OBTENIDOS				
	Comp. de acuerdo	De acuerdo	Ni de acuerdo, ni en desacuerdo	En desacuerdo	En completo desacuerdo
La aplicación ECMRE colabora en el aprendizaje de los conceptos presentados.	36%	57%	7%	0%	0%
Es útil contar con una herramienta de aplicación práctica que permita mediante ejemplos concretos visualizar los contenidos teóricos aprendidos en clase.	55%	38%	7%	0%	0%
Los contenidos en ECMRE se encuentran organizados y su utilización es intuitiva.	24%	52%	24%	0%	0%
La iconografía y gráficos utilizados en la aplicación tienen el tamaño adecuado y coinciden con la función asociada.	48%	38%	14%	0%	0%
Es beneficioso contar con esta aplicación como complemento a las clases teóricas.	52%	36%	12%	0%	0%

6 Conclusiones

Los temas de heterogeneidad, consumo energético y temperatura en arquitecturas paralelas son de gran importancia y se ha presentado una extensión de CMRE para incluirlos (ECMRE).

ECMRE aparece como una herramienta muy útil para la introducción de estos conceptos en los cursos iniciales de las carreras de Informática. Para lograr esto, se han realizado dos etapas, por un lado, la modificación del entorno CMRE para permitir manejar estas características en los robots y, por otro, se incorporaron herramientas gráficas para poder visualizar y posteriormente analizar esta información por parte de los alumnos de forma sencilla y amena.

De esta manera, el nivel de complejidad de los posibles escenarios aumenta, planteando un desafío mucho más ambicioso, que responde a la realidad tecnológica de los procesadores actuales.

Referencias

1. Hoare C. “Communicating Sequential Processes”. Prentice Hall, 1985.
2. Dijkstra E. W. “Finding the Correctness Proof of a Concurrent Program”. In Program Construction, International Summer Schoo, Friedrich L. Bauer and Manfred Broy (Eds.). Springer-Verlag, 24-34, 1978.
3. Hansen P. B. “The Architecture of Concurrent Processes”. Prentice Hall, 1977.
4. Dasgupta S. “Computer Architecture. A Moder Synthesis. Volume 2: Advanced Topics”. Jhon Wilet & Sons. 1989.
5. ACM Curriculum Committee on Computer Science. “Curriculum ‘68: Recommendations for the undergraduate program in computer science”. Communications of the ACM, 11(3):151-197. 1968.
6. ACM Curriculum Committee on Computer Science. “Curriculum ‘78: Recommendations for the undergraduate program in computer science”. Communications of the ACM, 22(3):147-166. 1979.
7. ACM Two-Year College Education Committee. “Guidelines for associate-degree and certificate programs to support computing in a networked environment”. New York: The Association for Computing Machinery. 1999.
8. Hoonlor A., Szymanski B. K., Zaki M. J., Thompson J. “An Evolution of Computer Science Research”. Communications of the ACM. 2013.
9. ACM/IEEE-CS Joint Task Force on Computing Curricula. “Computer Science Curricula 2013”. Report from the Task Force. 2013.
10. ACM/IEEE-CS Joint Task Force on Computing Curricula. “Computer Engineering 2004: Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering”. Report in the Computing Curricula Series. 2004.
11. ACM/IEEE-CS Joint Interim Review Task Force. “Computer Science Curriculum 2008: An Interim Revision of CS 2001”. Report from the Interim Review Task Force. 2008.
12. De Giusti, L., Leibovich, F., Sanchez, M., Chichizola, F., Naiouf, M., De Giusti, A. “Desafíos y herramientas para la enseñanza temprana de Concurrencia y Paralelismo”. Congreso Argentino de Ciencias de la Computación (CACIC), 2014.
13. AMD. “Evolución de la tecnología de múltiple núcleo”. http://multicore.amd.com/es-ES/AMD-Multi-Core/resources/Technology-Evolution. 2009.
14. De Giusti, L., Leibovich, F., Chichizola, F., Naiouf, M., De Giusti, A. “Incorporando conceptos en la enseñanza de Concurrencia y Paralelismo utilizando el entorno CMRE”. Congreso Argentino de Ciencias de la Computación (CACIC), 2015.
15. Gepner P., Kowalik M.F. “Multi-Core Processors: New Way to Achieve High System Performance”. In: Proceeding of International Symposium on Parallel Computing in Electrical Engineering 2006 (PAR ELEC 2006). Pags. 9-13. 2006.
16. Mc Cool M, “Programming models for scalable multicore programming”, 2007, http://www.hpcwire.com/features/17902939.html.
17. Balladini J., Rucci E., De Giusti A., Naiouf M., Suppi R., Rexachs D., Luque E. “Power Characterisation of Shared-Memory HPC Systems”. Computer Science & Technology Series–XVIII Argentine Congress of Computer Science Selected Papers. Págs. 53-65. 2013.
18. Brown D. J., “Toward Energy-Efficient Computing”, Magazine Communications of the ACM Volume 53 Issue 3, March 2010
19. De Giusti, A., De Giusti L., Leibovich, F., Sanchez, M., Rodriguez Eguren, S. “Entorno interactivo multirrobot para el aprendizaje de conceptos de Concurrencia y Paralelismo”. Congreso Tecnología en Educación, Educación en Tecnología. 2014.
20. Champredonde R., De Giusti A. “Herramienta visual para la enseñanza de programación”. Congreso Argentino de Ciencias de la Computación (CACIC), 1996.
21. Champredonde R., De Giusti A. “Design and Implementation of the Visual DaVinci Language”. Congreso Argentino de Ciencias de la Computación (CACIC), 1997.
22. https://nodejs.org/api/http.html