Control de velocidad media en tramos de carretera usando HPC

Procesamiento paralelo en tiempo real

Facultad de Ingeniería, Universidad de la República - Montevideo, Uruguay - Diciembre de 2012
Gustavo De Martino gustavodemartino@live.com

***Resumen*— La velocidad media de los automóviles en carretera puede determinarse utilizando los sistemas de captura de imágenes desarrollados para detección de velocidad instantánea. Comparando las imágenes tomadas entre dos puntos de control separados varios kilómetros, es posible determinar en tiempo real si un vehículo cometió una infracción en el tramo. El presente trabajo analiza el problema, detecta las limitaciones y evalúa una posible solución aplicando paralelismo para realizar las comparaciones de imágenes en tiempo real.**

Palabras clave— tiempo real, velocidad media, comparación de imágenes, redes neuronales, seguridad vial

I. INTRODUCCIÓN

El informe de Siniestralidad Vial en Uruguay de 2011 [1], presentado a mediados del año 2012 por la Unidad Nacional de Seguridad Vial (UNASEV), anuncia la penosa suma de 28.399 lesionados y 572 muertos en siniestros de tránsito en el último año, en los escasos 176.215 km² de la República Oriental del Uruguay. Menos agradable resulta el índice de 17 muertos cada 100.000 habitantes que supera ampliamente los valores registrados en la región y posiciona a este país en los primeros lugares del mundo. El 46.9% de los fallecimientos ocurrieron en rutas nacionales, donde la velocidad fue la causa directa o indirecta de los siniestros. Variadas son las medidas que se aplican para evitar estos "accidentes". La experiencia internacional muestra que la sanción es la medida preventiva más eficaz.

Los controles de velocidad puntuales son una solución cuando se implementan en forma masiva, pero esto no es impracticable en largos tramos de ruta.

Una idea alternativa, la que considera este trabajo, es controlar la velocidad media. Para esto es necesario detectar la presencia de un automotor, en dos lugares distintos ubicados a una distancia conocida. La velocidad promedio es siempre menor a la velocidad máxima, por lo que controlarla y sancionar los excesos sobre la misma resulta una medida muy adecuada para alcanzar el cambio de conducta deseado.

II. DESCRIPCIÓN DEL PROBLEMA

"El cambio de conducta en los conductores puede lograrse únicamente a través de controles e infracciones [1] ". La fiscalización realizada por inspectores es costosa y su alcance es muy limitado. Los equipos detectores de velocidad fijos, tienen elevado costo de instalación y mantenimiento puesto que los sensores se instalan bajo el pavimento y los sistemas deben calibrarse en períodos cortos de tiempo para asegurar su fiabilidad. Por otra parte, la señalización de los puntos de control genera conductas correctas de los conductores sólo en su entorno. Por estos motivos, colocar gran cantidad de puntos de control cercanos no es económicamente viable.

Una de las técnicas que se está comenzando a implementar en algunos países, es la identificación de la matrícula usando técnicas de OCR y la evaluación entre dos puntos de control alejados. Esto requiere imágenes de alta calidad y estructura uniforme en las matrículas. Por el momento no genera buenos resultados con imágenes nocturnas ni en zonas donde circulan automotores de distintos países.

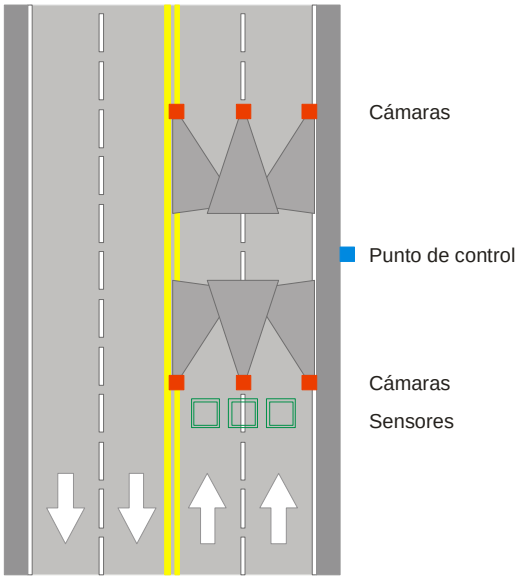

Figura 1 Esquema de un punto de control

La alternativa que presenta este trabajo no requiere identificación unívoca del automotor, sino coincidencia entre los registros de dos puntos de control consecutivos, pero requiere que las imágenes

[1] Palabras del Director de la Unidad Nacional de Seguridad Vial en la presentación del informe de Sinistralidad Vial en Uruguay de 2011

sean trasladadas de un punto de control al siguiente para ser comparadas. Por otra parte, cuanto mayor sea la distancia entre los controles, mayor será el tiempo, la cantidad de imágenes trasmitidas, almacenadas y comparadas.

III. USO DE PROCESAMIENTO PARALELO

El escenario para el que se pretende encontrar una solución es el de una ruta de dos carriles por sentido de circulación, donde los puntos de control están separados al menos diez kilómetros y el límite de velocidad sea de 110 Km/h.

Los automóviles deberían mantener una distancia entre sí que les permita detenerse ante una situación de emergencia. Esta distancia suele calcularse a partir del tiempo de reacción y la distancia de frenado. A los efectos de este trabajo, consideramos que en una ruta cargada, la distancia media entre automóviles es la mínima recomendada calculada a partir de un tiempo de reacción de 0.75 s y una desaceleración de 21 m/s², (valor calculado a partir de la "Guía para la conducción segura" [2])

Utilizando estos valores y considerando que nos interesa que el sistema sea capaz de soportar el peor caso, podemos calcular el flujo máximo de automóviles.

Para aprovechar la infraestructura existente, los puntos de control deben ser tan simples como sea posible.

La necesidad de utilizar los sistemas instalados, impone además algunos requisitos. Cada punto de control usa sensores bajo el pavimento y hasta seis cámaras para cada sentido de circulación como se muestra en la figura 1. El pasaje de un automóvil genera dos secuencias, cada una de cuatro a seis imágenes de alta resolución, una secuencia de frente y otra de atrás. Las imágenes de cada secuencia, están ligeramente desfasadas una de otra para adaptarse a las distintas velocidades de los automotores y no se puede predecir cuales son útiles y cuales no, tal como se muestra en la figura 2.

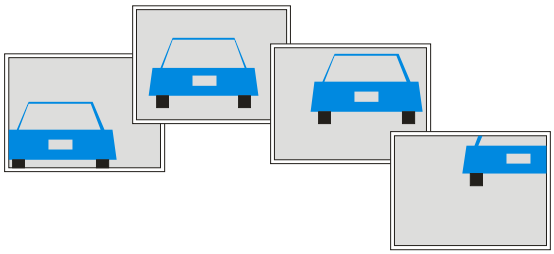

Figura 2 Secuencia de imágenes de un evento

Para la velocidad máxima considerada, el flujo esperado es de 0.91 automóviles por segundo en dos carriles.

Las fotografías tienen resolución de 5 Mp, cada imagen comprimida ocupa alrededor de 2.4 MB, lo que implica 28.8 MB de datos por automóvil.

Si consideramos como evento al conjunto de imágenes de un vehículo junto con la metadata asociada, el problema es por lo tanto encontrar concordancia entre los eventos de dos vectores infinitos, acotados a una ventana definida por dos velocidades cota y la distancia entre los puntos de control.

Si D es la distancia entre los puntos de control PC_1 y PC_2, en ese orden en el sentido del tráfico, V_i es la velocidad promedio para el tramo a partir de la cual se considera una infracción y V_m la máxima velocidad que un automotor podría desarrollar entre los puntos, entonces las cotas temporales para determinar que eventos deben compararse serán:

$$T_{min} = D / V_m$$

$$T_{max} = D / V_i$$

Para cada evento de PC_2 debe buscarse una correspondencia con todos los eventos de PC_1 cuyas marcas de tiempo se encuentren entre T_{min} y T_{max}.

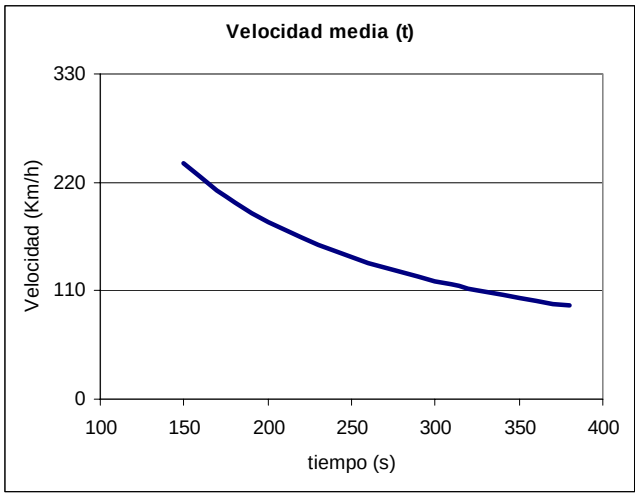

Figura 3 Relación velocidad – tiempo

Entre los parámetros del escenario objetivo, fijamos el valor de V_m en 220 Km/h, lo que define una ventana de 164 segundos entre T_{min} y T_{max}. Por lo tanto cada evento de PC_2 deberá compararse hasta con otros 148 de PC_1. Si consideramos que comparar dos eventos requiere realizar 72 comparaciones entre imágenes, necesitaremos realizar unas 9500 comparaciones por cada automóvil detectado en PC_2, es decir unas 10.500 comparaciones por segundo.

Algoritmo de comparación

La versión evaluada del comparador es una red neuronal de tres capas. En la primera hay una neurona por cada área de 1024 pixeles pertenecientes a la primera fotografía (32x32).

Cada neurona tiene como entradas esos pixeles y otros 4096 de un área de la segunda fotografía alrededor de las coordenadas de la primera.

La evaluación de cada entrada requiere de 3 operaciones de punto flotante. Las operaciones de la segunda y tercera capa son despreciables.

Este proceso requiere alrededor de $8x10^6$ operaciones de punto flotante para cada comparación, alrededor de 83 GFLOPS [2], capacidad de cálculo difícil de alcanzar con equipos de propósito general. De allí la necesidad de investigar la alternativa en procesamiento paralelo para proveer una solución de bajo costo.

IV. ESTRATEGIA DE LA SOLUCIÓN

Suponiendo que la capacidad de cómputo no sea una limitante, el proceso de comparación necesita recibir dos flujos de datos de alrededor de 210 Mbps. Como se puede observar en la figura 3, la mitad de estos datos -los provenientes del primer punto de control en el sentido de circulación- serán utilizados luego de transcurrido un tiempo entre T_{min} y T_{max}. Para el caso que estamos considerando, 327 segundos de datos de un punto de control equivalen a 8.35 GB, espacio de memoria que se necesitará para almacenar las imágenes recibidas mientras tengan que realizarse las comparaciones.

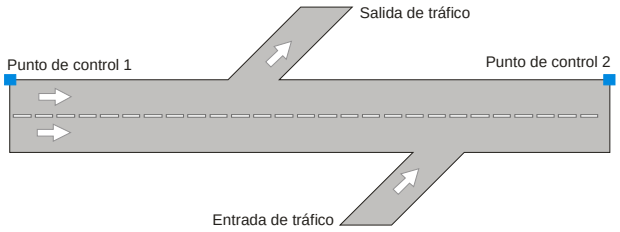

Figura 4 Escenario considerado

Los flujos de imágenes de dos puntos de control consecutivos deben compararse en un lugar físico. Ya que no hay diferencia entre los flujos, debe existir un enlace entre los puntos de control capaz de trasladar al menos 210 Mbps de datos.

El la figura 5 se muestra un esquema de la correlación entre eventos de uno y otro punto de control. Cada evento de PC_2, deberá compararse con un conjunto de eventos de PC_1. El siguiente evento de PC_2, deberá compararse con un conjunto similar por lo que es muy conveniente que estas tareas sean realizadas por el mismo procesador para evitar transferencias de datos innecesarias.

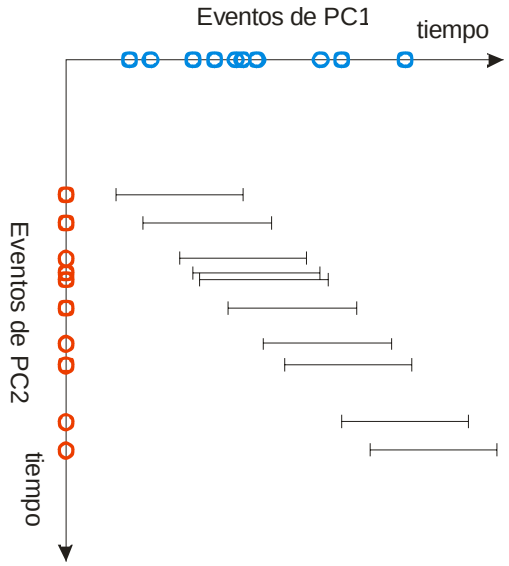

Figura 5 Correlación de eventos

Algunos eventos de PC_1, deben ser evaluados en más de un procesador y es posible que un procesador no sea capaz de retener todos los eventos de PC_1 contra los que debe evaluarse cada uno de PC_2. Se necesita por lo tanto o bien un proceso distinguido que se ocupe de mantener el control de esos datos, o bien un trabajo colaborativo entre los procesos que realizan las comparaciones.

Lógica de los puntos de control

Para que las marcas de tiempo sean válidas, los relojes de los puntos de control deben estar perfectamente sincronizados. De no ser así, el error debe considerarse en el momento de calcular las cotas temporales para determinar el entorno temporal en el que se detecta una infracción.

Las imágenes obtenidas deben almacenarse en un buffer mientras sea necesario realizar una evaluación y hasta que puedan ser útiles para detectar una posible coincidencia.

Si las condiciones del tráfico hacen que el buffer se desborde, deben descartarse los elementos más antiguos ya que es más probable que hayan sido evaluados.

Cada punto de control tiene que realizar las siguientes actividades:

- Captura y registro de los eventos
- Sincronización de hora
- Entrega de datos (descarga de buffer)

Lógica del comparador

El comparador debe implementarse sobre un sistema capaz de proveer la capacidad de cómputo necesaria para realizar todas las comparaciones del escenario objetivo.

[2] Datos provistos por BOLDT S.A. Argentina, empresa que está desarrollando el comparador.

Los eventos en memoria -imágenes y metadata- son de solo lectura, lo que permite usar procesos concurrentes sin introducir mayor complejidad.

Sobre este sistema, se ejecutaran tantas instancias del programa que realiza la comparación de eventos, como sea posible para maximizar la cantidad de comparaciones.

Para maximizar el rendimiento, es necesario minimizar las transferencias de datos por lo que cada instancia del programa alojará en su memoria local tantos eventos de PC_1 consecutivos en el tiempo como sea posible para luego recibir uno a uno los eventos de PC_2 contra los que debe comparar.

En algunos casos, será necesario que más de una instancia obtenga el mismo evento de PC_2. Para esto, el proceso que los recibe, informa a todos los comparadores del nuevo evento y su marca de tiempo.

Cuando se encuentra una coincidencia, los eventos deben enviarse a un proceso para que lo almacene o reporte según corresponda.

Un sistema con memoria compartida y múltiples hilos, es por lo tanto el ambiente más adecuado para implementar la solución.

Balance de carga

Los datos del primer punto de control en el sentido del tráfico deben retenerse hasta que la marca de tiempo del segundo punto de control supere en T_{max} a la marca de tiempo de los eventos.

El comparador que retiene estos datos debe descartarlos para solicitar nuevos elementos.

La distribución que propone el punto anterior, determina que los datos de PC_1 sean entregados a un comparador hasta que este no tenga más espacio libre.

Si bien esto resulta en una distribución circular, la carga de los procesadores no se mantiene balanceada sino que los datos se van distribuyendo a medida que los procesadores quedan libres y uno de los procesadores -el que está renovando datos- queda potencialmente ocioso la mitad del tiempo.

Tolerancia a fallos

El sistema comparador deberá instalarse en un lugar cercano a alguno de los puntos de control para minimizar los costos de comunicaciones ya que las transferencias de datos son importantes.

No hay razones para distribuir geográficamente los procesadores por lo que no se esperan fallas en las comunicaciones o los equipos. Los procesos son deterministas y el sistema no es crítico por lo que no se justifica incrementar la cantidad de recursos de procesamiento para ofrecer tolerancia a fallos.

V. EVALUACIÓN EXPERIMENTAL

La implementación se ha realizado utilizando MPI para la distribución de datos y threads para realizar las comparaciones. Para las pruebas se utilizó un cluster con procesadores Xeon 5500 W5580 "Gainestown" de 4 núcleos y 3.20 GHz con 8 GB de RAM[3].

Como referencia se implementó un algoritmo serial sobre un núcleo de este procesador y se trabajó sobre una muestra de 10 minutos de tráfico a flujo máximo (544 eventos, 6528 imágenes, 15.3 GB). En todos los casos se introdujo una espera de 320 segundos antes de iniciar los comparadores para que el tiempo de generación y la distribución inicial de los datos no incidieran en la medición del tiempo de procesamiento.

Simulación de tráfico

Para poder verificar la implementación y evitar el manejo del importante volumen de datos que esta requiere, se utilizó un simulador de tráfico con las siguientes características:

- Entre dos puntos de control pueden haber tanto ingresos como egresos de vehículos.
- Todos los vehículos que circulan por un carril en un momento dado tienen un comportamiento similar.
- Los vehículos que circulan entre los puntos de control salen de la vía con una probabilidad conocida.
- Otros vehículos ingresan a la vía por carreteras auxiliares.
- Los elementos que diferencian a los vehículos están dados por:
    - El color.
    - La forma física (consecuencia del conjunto marca-modelo y las personalizaciones entre las que se encuentra la matrícula)

En la figura 4 se muestra un esquema del escenario considerado. Allí se puede observar que no todos los eventos de PC_1 tendrán una coincidencia con alguno de PC_2. También queda claro, que los eventos de PC_2 no necesariamente tendrán un correspondiente entre los datos provenientes de PC_1.

Implementación del simulador

Cada vehículo se representa con un identificador y un conjunto de coordenadas en un espacio de

[3] Gentileza de Documentos de Seguridad BOLDT (Dorebor S.A.)

dimensión tres, restringido a una bola de diámetro 1, de modo que la distancia entre dos puntos de ese espacio sea un valor entre 0 y 1 equivalente a uno menos la probabilidad de que sus imágenes sean iguales.

Las fotografías de un vehículo en un punto de control están desfasadas una respecto a la otra. Esta fase puede expresarse como un valor entre cero y uno; cero cuando el vehículo está totalmente fuera de la fotografía, uno cuando está totalmente centrado.

El algoritmo de comparación, a los efectos de este trabajo se considerará una caja negra que recibe dos imágenes y devuelve un valor entre 0 y 1, donde 0 significa "totalmente diferente" y 1 "correspondencia total".

Al comparar dos fotografías de un mismo vehículo con un comparador con un error "e" tomadas respectivamente en los puntos de control PC_1 y PC_2, con fases f_1 y f_2 se obtiene un resultado: $f_1 * f_2 \pm e$

Si las fotografías pertenecen a dos vehículos cuya distancia en la representación es ***d***, el resultado de la comparación será $min(1, (1-d) * f_1 * f_2 \pm e)$

Aceleración esperada

El algoritmo se diseñó para minimizar las comunicaciones y evitar que estas incidan en el tiempo total de procesamiento. Las actividades seriales son casi despreciables frente al proceso de comparación de imágenes, pero el flujo de datos puede transformarse en un impedimento. Los procesos que reciben los datos de los puntos de control y los envía a los comparadores deben ser capaces de manejar simultáneamente flujos de entrada y salida. Si los algoritmos de comparación no son eficaces, el canal de salida puede saturarse. Si las comunicaciones o las interfases de red no son adecuadas, el éxito puede resultar comprometido.

Escalabilidad del problema

Incrementar la cantidad de carriles aumenta proporcionalmente el flujo de datos. El escenario actual requiere flujos por encima de 200 Mbps a lo largo de 10 Km. lo que implica enlaces de fibra óptica, por lo que el incrementar el ancho de banda de salida de los puntos de control no genera mayores problemas.

Incrementar la distancia entre los puntos de control aumenta linealmente la cantidad de memoria requerida, potencialmente la cantidad de comparaciones y con el mismo orden necesidad de procesamiento. Aunque esto puede solucionarse agregando procesadores, las comunicaciones entre procesos puede verse comprometida.

La tabla 1 muestra la necesidad de cómputo teórica, expresada en cantidad de núcleos de referencia en función de la cantidad de carriles y la distancia entre los puntos de control considerando aceleración lineal.

El algoritmo escala linealmente con la distancia y potencialmente con la cantidad de carriles (aproximadamente $12n^2$ con n la cantidad de carriles para el escenario considerado).

Tabla 1 Escalabilidad de la aplicación

Núcleos de referencia necesarios		Carriles			
		2	**3**	**4**	**5**
Distancia (Km)	**5**	23	53	94	146
	10	47	105	187	292
	20	94	211	374	585

Estrategia de paralelización

Un sistema paralelo puede aportar tanta capacidad de cómputo como sea necesario, pero genera necesidades de comunicación.

Como se mencionaba anteriormente, cada evento del PC_2, debe compararse con todos los eventos de PC_1 que se encuentran en la franja de tiempo entre $T_{actual} - T_{max}$ y $T_{actual} - T_{min}$, donde T_{actual} es la marca de tiempo del evento en PC_2.

Para el escenario analizado, estos datos ocupan algo más de 8 GB, que podrían alojarse en la memoria asociada a un solo núcleo de procesamiento. Para generalizar la solución, de modo que permita por ejemplo ser válida para escenarios con una distancia mayor entre los puntos de control, se consideró que estos datos no necesariamente puedan alojarse en la memoria asociada a un núcleo.

El flujo de datos proveniente de cada punto de control, recorre un camino físico hasta la interfaz de red de un equipo. El proceso que recibe los datos del primer punto de control en la dirección de circulación, debe hacer la distribución de estos y por lo tanto resulta distinguido. Por otra parte, el flujo de datos proveniente del segundo punto de control en la dirección de circulación alcanza a un segundo proceso, el que debe entregar los datos a los procesos que tienen los eventos contra los que se debe comparar. Esto lo transforma en un segundo proceso distinguido.

No es razonable que estos dos flujos de datos converjan en un único equipo, porque forzarían una topología compleja para evitar que el canal de comunicación se convierta en un cuello de botella. Tampoco es razonable almacenar los datos en disco porque esto duplicaría el flujo de información imponiendo mayores restricciones.

Por último, los resultados positivos de las comparaciones deben canalizarse hacia algún destino, junto con la evidencia que hasta el momento no fue almacenada. Este proceso puede estar físicamente cerca del sistema de archivos o tener acceso a un enlace dedicado a transmitir estos resultados.

El peor caso es aquel en que todas las comparaciones resulten positivas, lo que sucedería con un comparador incorrecto o un conjunto de automóviles idénticos. Para este caso el algoritmo diseñado generaría un desbordamiento en los distribuidores de entrada.

Un caso crítico pero factible, es que todos los vehículos excedan el límite de velocidad. En esta situación y considerando el escenario analizado, el comparador de salida estaría manejando flujos de entrada y salida simultáneos de 420 Mbps de datos cada uno, valor cercano al límite que puede manejar una interfaz de red de 1 Gbps.

Deben existir por lo tanto tres procesos distinguidos que se ejecutan en equipos distinguidos y un conjunto de procesadores que se ocupan de las comparaciones.

Procesos y mensajes

Los procesos distinguidos no tienen requerimientos especiales de memoria o procesamiento, por lo que los equipos dedicados a estos fines, pueden también actuar como comparadores.

Con estas consideraciones, todos los procesos son comparadores y durante la inicialización, verifican si -de acuerdo al hardware- deben ejecutar además algún proceso distinguido.

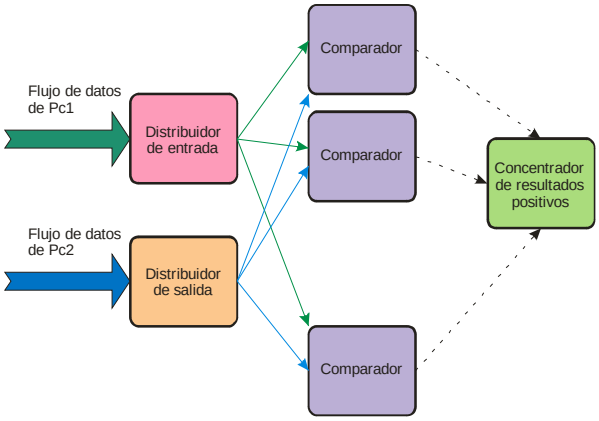

Figura 6 Procesos y flujo de datos

Los procesos distinguidos se ejecutan como nuevos hilos. En la

figura 6 se muestra la lógica de la comunicación.

A continuación se muestra un pseudocódigo de la inicialización. El proceso es análogo para determinar cuáles son los identificadores del distribuidor de salida y el concentrador de salida. Nótese que la distribución de la identidad no puede realizarse con la primitiva Broadcast porque sólo el proceso en el equipo distinguido conoce el identificador del root.

```
my_rank = get_rank()
if ( is_input_distributor() )
    new_thread( input_distributor )
    input_distrubutor_id = my_rank
    send_to_all_other( my_rank )
else
    input_distrubutor_id = read()
```

En la figura 7 la comunicación entre el distribuidor de entrada y los comparadores. Nótese respecto al balance de carga, que hasta que el comparador que está recibiendo datos no se declara completo, el distribuidor no entrega eventos a otro.

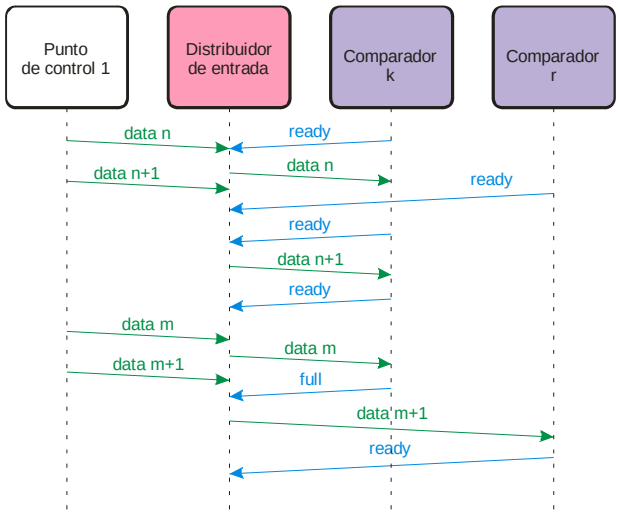

Figura 7 Distribución de datos de PC_1

En la figura 8 la comunicación entre el distribuidor de salida y los comparadores.

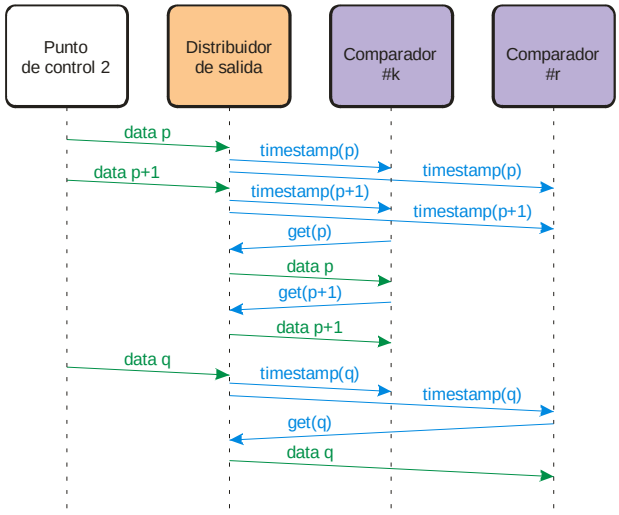

Figura 8 Distribución de datos de PC_2

Existe un mensaje adicional que no se muestra en el diagrama. Este mensaje es enviado por el

distribuidor de salida a todos los comparadores indicando una la marca de tiempo para que se descarten eventos no vigentes (útil para el caso en que se corte temporalmente el flujo de vehículos a través del segundo punto de control).

Una vez que un comparador recibe un evento del distribuidor de salida, ejecuta un nuevo hilo para cada comparación.

Los posibles positivos, donde el comparador devuelve algún resultado por encima de una cota de referencia, son enviados al concentrador de salida.

Resultados de las pruebas

La aceleración del algoritmo se define según la siguiente fórmula:

$$S_p = \frac{T_1}{T_p}$$

donde:

- p es el número de procesadores
- T1 es el tiempo de ejecución del algoritmo serial
- Tp es el tiempo de ejecución del algoritmo paralelo con p procesadores

La eficiencia es una medida del rendimiento definida como:

$$E_p = \frac{S_p}{p} = \frac{T_1}{pT_p}$$

El algoritmo está diseñado para operar en tiempo real. Esto implica que cuando los flujos de entrada exceden las capacidades de procesamiento, los datos se descartan.

Para realizar las medidas de rendimiento, se modificó el generador de datos de entrada (simulador de tráfico) para que entregue una cantidad limitada de eventos y con un flujo constante. Para cada caso el flujo se calibró para que la cantidad de elementos descartados por desbordamiento no excediera el 2% del total.

El flujo analizado es el equivalente a 600 segundos (10 minutos) al flujo máximo esperado. Bajo estas condiciones se realizaron las medidas que se encuentran en la tabla 3. La ejecución serial de referencia requirió un promedio alrededor de 7 horas 48 minutos.

Tabla 2 Tiempos de ejecución

Algoritmo	Equipos #	Núcleos #	Ejecuciones #	Duración (s)	Aceleración Sp	Eficiencia Ep	Rendimiento
Serial	1	1	3	28071	-	-	2%
Multithread	1	4	6	7112	3.9	0.99	8%
Multithread	1	8	5	3631	7.7	0.97	17%
Paralela	2	16	6	1882	14.9	0.93	32%
Paralela	8	64	6	578	48.6	0.76	104%

La aceleración del algoritmo se representa en el gráfico de la figura 9 contrastado con la situación ideal conocida como aceleración lineal (Sp = p).

La tabla 2 contiene un resumen de los resultados. Incluye además el campo “Rendimiento”, como relación entre el tiempo esperado y el tiempo de ejecución, útil para determinar la capacidad de cómputo necesaria para implementar el sistema de tiempo real.

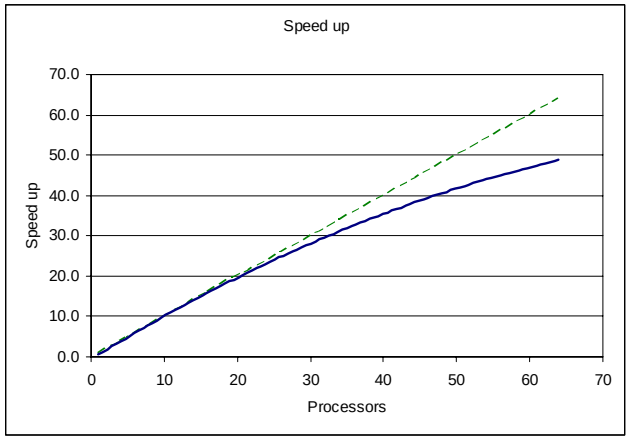

Figura 9 Speed up

VI. ESCALABILIDAD

De la eficiencia resultante se concluye que la porción serial equivalente del algoritmo es de

alrededor de 0.5% calculado en base a la Ley de Amdahl para la implementación paralela.

Fueron necesarios 64 núcleos para alcanzar los requerimientos del escenario planteado. Si bien se optimizaron las comunicaciones, el resultado no es muy cercano a una situación lineal, lo que compromete la escalabilidad.

Un escenario con cuatro carriles, requeriría una capacidad de procesamiento mucho mayor que cuestionaría la viabilidad de la instalación.

Es posible minimizar los requerimientos, pero de todos modos la solución resulta limitada.

Una mejora a considerar es la de aumentar drásticamente la capacidad de cómputo utilizando GPU. De todos modos, luego de duplicar el volumen de datos respecto al escenario analizado, los canales de comunicación de 1 Gbps comenzarán a resultar comprometidos.

VII. CONCLUSIONES

El algoritmo desarrollado resulta útil para el escenario, muy especialmente porque utilizan la infraestructura existente.

Debe considerarse cuidadosamente la extrapolación a otros escenarios. Para ello es recomendable obtener datos precisos acerca del flujo de automóviles. La distancia entre puntos de control puede aumentarse esperando un aumento en los requerimientos lineal con la distancia. La cantidad de carriles explorados puede aumentarse de dos a cuatro esperando un aumento cuadrático en los requerimientos de cómputo.

VIII. REFERENCIAS

[1] “Siniestralidad vial en Uruguay 2011”, Unidad Nacional de Seguridad Vial, Publicado por Presidencia de la República, Uruguay, Julio 2012. Disponible en: http://archivo.presidencia.gub.uy/unasev/news/2012/Informe 2011.pdf Visitado en diciembre 2012.

[2] “Guía para la conducción segura”, Intendencia Municipal de Montevideo, Uruguay. Publicado en noviembre de 2012. Disponible en: “http://www.montevideo.gub.uy/sites/default/files/articulo/guia_para_la_conduccion_segura_2.pdf”. Visitado en diciembre de 2012.

[3] “MPI: The Complete Reference”. Marc Snir, Steve Otto, Steven Huss-Lederman, David Walker, Jack Dongarra. MIT Press - 1995. Disponible en http://www.netlib.org/utk/papers/mpi-book/mpi-book.html, visitado en diciembre de 2012.

[4] “Redes neuronales analógicas para la computación paralela masiva de señales en tiempo real”, Cancelo, Gustavo Indalecio Eugenio, Laboratorio de Electrónica Industrial, Control e Instrumentación (LEICI), publicado en 1996, disponible en http://biblioteca.universia.net/html_bura/ficha/params/id/55240408.html, visitado en diciembre de 2012.

IX. ANEXO - DATOS EXPERIMETNALES

Tabla 3 Detalle de las pruebas

Algoritmo	Equipos	Núcleos (p)	Período (ms)	Eventos procesados	Eventos descartados	Desbordamiento	Duración (h:mm:ss)	Duración (s)	Aceleración Sp	Eficiencia Ep
Serial	1	1	47800	535	9	1.7%	7:50:25	28225	**0.99**	**0.99**
Serial	1	1	47500	534	10	1.8%	7:47:04	28024	**1.00**	**1.00**
Serial	1	1	47300	538	6	1.1%	7:46:04	27964	**1.00**	**1.00**
Multithread	1	4	12400	536	8	1.5%	1:59:41	7181	**3.91**	**0.98**
Multithread	1	4	12200	535	9	1.7%	1:59:26	7166	**3.92**	**0.98**
Multithread	1	4	12095	534	10	1.8%	1:58:46	7126	**3.94**	**0.98**
Multithread	1	4	12000	538	6	1.1%	1:57:51	7071	**3.97**	**0.99**
Multithread	1	4	12000	539	5	0.9%	1:57:45	7065	**3.97**	**0.99**
Multithread	1	4	11800	543	1	0.2%	1:57:43	7063	**3.97**	**0.99**
Multithread	1	8	6200	534	10	1.8%	1:01:14	3674	**7.64**	**0.96**
Multithread	1	8	6180	538	6	1.1%	1:00:32	3632	**7.73**	**0.97**
Multithread	1	8	6160	536	8	1.5%	1:00:13	3613	**7.77**	**0.97**
Multithread	1	8	6140	533	11	2.0%	0:59:50	3590	**7.82**	**0.98**
Multithread	1	8	6120	542	2	0.4%	1:00:48	3648	**7.69**	**0.96**
Paralela	2	16	3220	533	11	2.0%	0:31:10	1870	**15.01**	**0.94**
Paralela	2	16	3200	538	6	1.1%	0:31:25	1885	**14.89**	**0.93**
Paralela	2	16	3180	540	4	0.7%	0:31:38	1898	**14.79**	**0.92**
Paralela	2	16	3160	533	11	2.0%	0:31:40	1900	**14.77**	**0.92**
Paralela	2	16	3140	533	11	2.0%	0:31:31	1891	**14.84**	**0.93**
Paralela	2	16	3120	536	8	1.5%	0:30:50	1850	**15.17**	**0.95**
Paralela	8	64	1050	543	1	0.2%	0:09:32	572	**49.08**	**0.77**
Paralela	8	64	1020	541	3	0.6%	0:09:43	583	**48.15**	**0.75**
Paralela	8	64	1000	541	3	0.6%	0:09:26	566	**49.60**	**0.77**
Paralela	8	64	1000	536	8	1.5%	0:09:27	567	**49.51**	**0.77**
Paralela	8	64	970	538	6	1.1%	0:09:54	594	**47.26**	**0.74**
Paralela	8	64	950	533	11	2.0%	0:09:46	586	**47.90**	**0.75**