UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Semestre 2024-1
Sistemas Operativos
Grupo 06

IMPACTO DE LA SINCRONIZACIÓN Y EL PARALELISMO EN LOS VIDEOJUEGOS

Ing. Gunnar Eyal Wolf Iszaevich

Alumnos:

- **R**odríguez **K**obeh **S**antiago
- **V**illaseñor **V**enegas **C**arlos **M**iguel

Introducción

Hoy en día los videojuegos son una de las industrias más populares y con mayor éxito, no solo en el mundo del entretenimiento, si no también dentro de la educación, ingeniería y estudio científico, llegando a superar las ganancias de la industria de la música y de las películas[1], es por esto que siempre andan en la búsqueda por nuevas y mejores formas de brindar un mejor producto. Los videojuegos al ser un producto que depende de múltiples disciplinas, se ve limitado por los avances tecnológicos disponibles, debido a ello la tendencia de los CPUs de aumentar el número de núcleos, en vez de la velocidad, hace que sea de suma importancia que se busque la forma de incorporar estos conceptos a la industria.

Conceptos motores de videojuegos

Motor de videojuegos: Es un entorno de trabajo diseñado principalmente para la realización de videojuegos.

Componentes de un motor de videojuegos: Partes de un motor de videojuegos que se encarga de una función en específico, los principales componentes son:

- Input
- Lógica del juego
- Inteligencia artificial
- Físicas
- Audio
- Gráficas 3D
- Renderizador

Paralelismo de datos

El paralelismo en los datos consiste en dividir los datos que se buscan procesar en varios hilos.

Este paradigma de programación concurrente se emplea dentro de los motores de videojuegos cuando un componente recibe información y este la divide en varios subprocesos que manda a hilos para ser ejecutados de manera concurrente. Cabe mencionar que aunque este proceso es concurrente, la ejecución de las demás instrucciones seguirá siendo

secuencial. Es importante mencionar que cuando se emplea este paradigma, se puede mejorar el agrupamiento de datos para que los que vayan a interactuar entre sí, estén en el mismo hilo.

Paralelismo de tareas

El paralelismo de tareas es otro paradigma de programación concurrente que consiste en distribuir diferentes procesos en diferentes hilos.

Este puede ser aplicado según dos tipos de modelos:

1. Síncrono: Este modelo consiste en ejecutar todas las tareas dentro de un mismo ciclo de reloj, para después ser repetidas desde el inicio nuevamente manteniendo el mismo orden de ejecución. Similar a estar dentro de un ciclo “while”.

2. Asíncrono: Este modelo consiste en que cada proceso se ejecuta independientemente dentro de su propio ciclo de reloj, lo cuál podría generar problemas, ya que las tareas que se realizan dentro de un videojuego suelen ser fuertemente dependientes unas de otras. Es por esto que se ocupa una etapa de sincronización entre cada ciclo. Este modelo es especialmente útil para componentes que no se requieren actualizar cada ciclo, como el input del usuario.

Lo mejor de ambos mundos

Al momento de decidir sobre cuál paradigma utilizar, es importante considerar que si uno decide utilizar el paralelismo de datos, el motor será muy bueno para realizar tareas sobre procesos que sean secuenciales, pero no aprovechará la concurrencia con tareas que se encuentren separadas, debido a las condiciones de carrera que se podrían generar. Esto hace difícil la decisión, ya que muchas veces, el motor se enfrentará a ambas situaciones tarde o temprano. Afortunadamente no tenemos que decidir.

Como ya vimos, cada uno de estos paradigmas tiene sus ventajas según la situación, es por ello que para que un motor de videojuegos se aproveche del paralelismo, debe de utilizar ambos paradigmas.

Balanceo de cargas

El balanceo de cargas consiste en distribuir el trabajo que realiza cada procesador de forma uniforme (Administrador de procesos). Un método muy comúnmente utilizado es el “thread pool”. Este método lo que hace es que para cada componente, agrega a una cola las tareas que están pendientes, de donde cada hilo que esté disponible tomará la siguiente tarea a realizar.

Sincronización

Aunque en muchos programas resulta muy conveniente el uso de “mutex” para encargarse de la sincronización, en los motores de videojuegos, este puede generar sobrecalentamiento, bloqueos mutuos e inversión de prioridad. Es por esto que se suele optar por otras estrategías como tener una etapa de sincronización donde todos los procesos deben de ejecutarse en secuencia, implementar algún método de comunicación entre hilos, o incluso usar un algoritmo que no use candados (Lock-free Algorithms), lo que conlleva una planificación más difícil, para evitar condiciones de carrera.

Modelos de implementación de motores de videojuegos

Combinando los diferentes conceptos mencionados, podríamos generar muchos modelos de implementación del paralelismo a un motor de videojuegos. A continuación mostraremos una comparación entre 3 opciones de implementación junto con un modelo secuencial realizada por el departamento de ingeniería eléctrica y ciencias de la computación de la universidad de Wichita [2] usando el juego “Tower Defense Game ” en un procesador 8 núcleos a 2.13 GHz y 6 GB de RAM.

- *Modelo de un hilo (STM)*: Este modelo consta de recibir algún input, y luego procesarlo a través de los distintos componentes de manera secuencial.
- *Modelo de múltiples hilos asíncronos (MAM)*: Este modelo divide en 2 hilos con su propio ciclo de reloj la responsabilidad sobre ciertos componentes.
- *Modelo de múltiples hilos síncrono (MSM)*: Este modelo usa un algoritmo libre de candados, ya que la sincronización es realizada dividiendo las tareas en secuencial y paralelo, la parte secuencial es la que evita que se trabaje con los componentes que podrían generar una situación de carrera en el caso de ser ejecutados en paralelo.

- *Modelo de múltiples hilos síncrono con paralelismo de datos (MSMDP)*: Este modelo trabaja de forma muy similar que el anterior modelo, con la diferencia de que el componente encargado de las físicas, al ser ejecutado, se dividirá en dos hilos para la detección de colisiones.

La primera prueba realizada consta en ver cuantos frames genera cada modelo en 3 minutos.

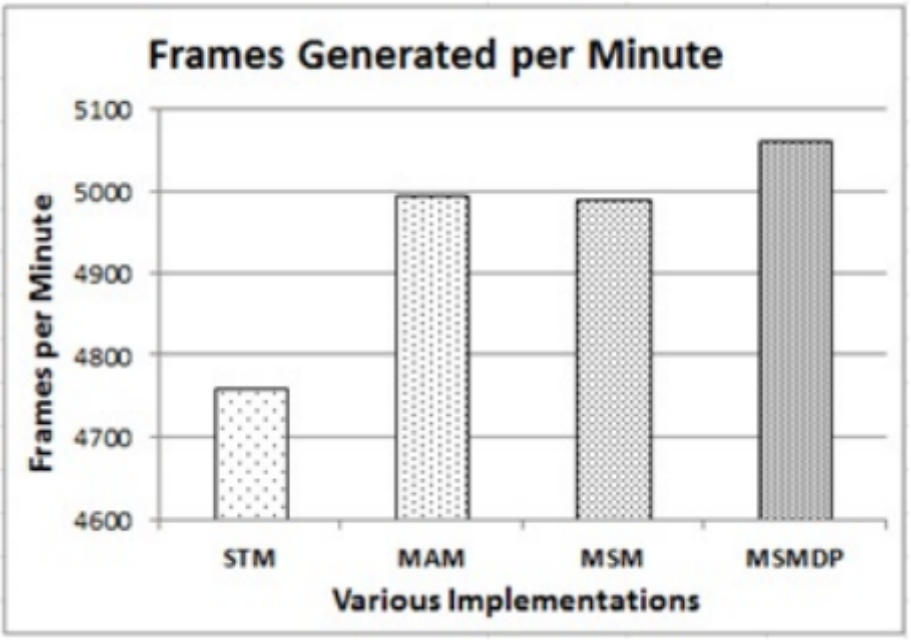

Fig. 1: Número de frames generados en un minuto por el juego "Tower Defense Game". Imagen obtenida de [2].

El segundo experimento consiste en mostrar el tiempo máximo que se requiere para procesar diferentes frames.

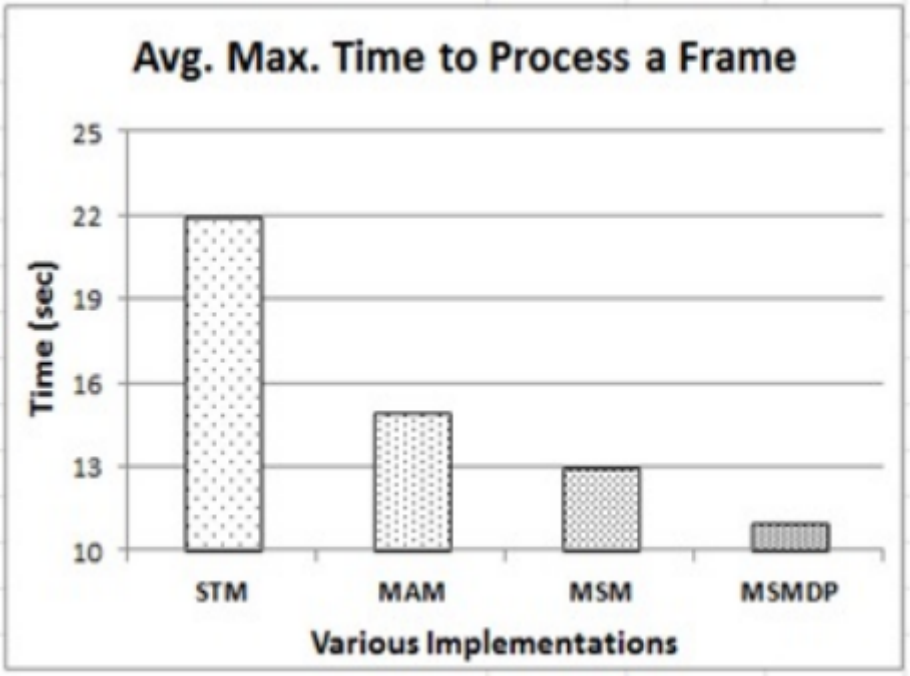

Fig. 2: Promedio del tiempo máximo de procesamiento para cada frame. Imagen obtenida de [2].

Podemos ver que el modelo secuencial y el modelo asíncrono son los que pueden llegar a tomar más tiempo en procesar cada frame. A diferencia del modelo MSMDP.

El tercer experimento consiste en mostrar el máximo tiempo requerido para procesar diferentes componentes.

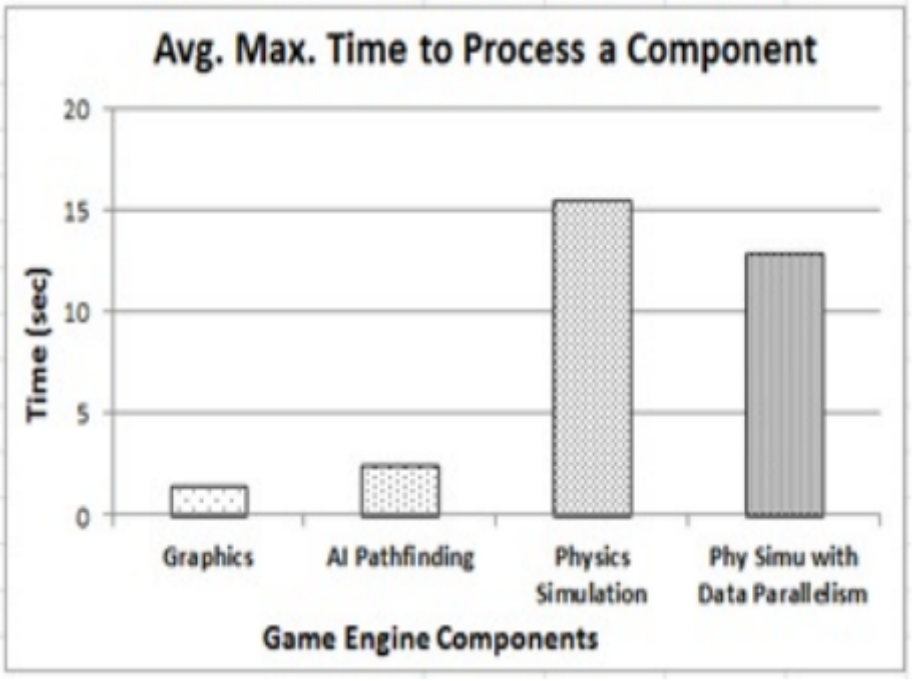

Fig. 3: El promedio del tiempo máximo de procesamiento de cada componente. Imagen obtenida de [2].

Podemos ver que el componente encargado de los físicas, es el que más tiempo consume, y que este se ve ligeramente mejorado por el uso de paralelismo de datos.

El cuarto experimento consiste en observar la ganancia obtenida de paralelizar con respecto a usar un único hilo.

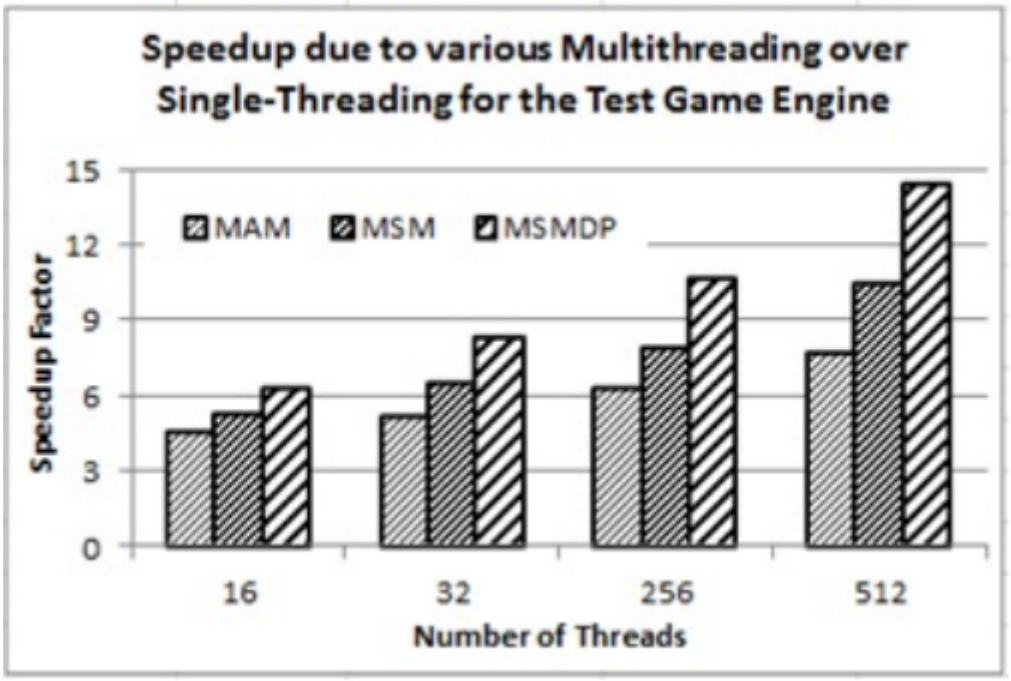

Fig. 4: Ganancia derivada de paralelizar. Imagen obtenida de [2].

El cuarto experimento nos muestra que la ganancia derivada de paralelizar va aumentando con el número de hilos, llegando a ser 14 veces mayor que el de un solo hilo.

Conclusiones de los experimentos

Con base en los resultados, se puede observar que el modelo con le mejor desempeño fue el modelo de múltiples hilos síncrono con paralelismo de datos, con esto podemos ver que el uso de multiprocesamiento puede ser una gran respuesta a las limitantes de hardware para mejorar el rendimiento de los motores de videojuegos.

Referencias

[1] Divers, G. (s. f.). *Gaming industry dominates as the Highest-Grossing Entertainment Industry | GamerHub*. Gamerhub. https://gamerhub.co.uk/gaming-industry-dominates-as-the-highest-grossing-entertainment-industry/#:~:text=According%20to%20recent%20data%2C%20the,%2C%20mobile%20games%2C%20and%20esports.

[2] Best, M. J., Fedorova, A., Dickie, R., Tagliasacchi, A., Couture-Beil, A., Mustard, C., ... & Brownsword, A. (2009). Searching for concurrent design patterns in video games. In Euro-Par 2009 Parallel Processing: 15th International Euro-Par Conference, Delft, The Netherlands, August 25-28, 2009. Proceedings 15 (pp. 912-923). Springer Berlin Heidelberg.

[3] Wikipedia contributors. (2023, October 22). Game engine. In *Wikipedia, The Free Encyclopedia*. Retrieved 20:39, October 28, 2023, from https://en.wikipedia.org/w/index.php?title=Game_engine&oldid=11813425888