Inciso 7: Análisis de los Acercamientos

Javier Valle, Mario de León

1 Descripción del Acercamiento Propuesto (DKSP)

El acercamiento "Dynamic Key Space Partitioning" (DKSP) tiene como objetivo mejorar la eficiencia de la búsqueda paralela de una clave en un sistema criptográfico. A diferencia del método "naive" de dividir estáticamente el espacio de claves entre procesos, DKSP asigna dinámicamente "lotes" de claves a los procesos de acuerdo con sus necesidades. Este método garantiza que todos los procesos permanezcan ocupados, minimizando así el tiempo de inactividad y mejorando la utilización de los recursos disponibles.

Algoritmo Descriptivo:

1. **Inicialización**: El proceso maestro (rank 0) divide el espacio total de claves en "lotes" de tamaño L.
2. **Distribución de Lotes**: Cada proceso, incluido el maestro, recibe un lote inicial de claves para procesar.
3. **Procesamiento de Claves**: Cada proceso cifra y descifra el texto con cada clave de su lote actual. Si un proceso encuentra la clave correcta, envía una señal a todos los demás procesos para terminar la búsqueda.
4. **Solicitud de Más Claves**: Cuando un proceso termina de procesar su lote actual, solicita un nuevo lote al proceso maestro.
5. **Finalización**: Si se encuentra la clave correcta o si se agotan todas las claves posibles, todos los procesos terminan.

2 Derivación del Valor Esperado de $t_{Par}(n, k)$

Para el enfoque DKSP, asumimos una distribución uniforme de la clave correcta y un sistema ideal sin latencia adicional causada por la comunicación o la gestión de lotes. Bajo estas condiciones, podemos aproximar el tiempo paralelo esperado $E[t_{Par}(n, k)]_{DKSP}$ como:

$$E[t_{Par}(n, k)]_{DKSP} \approx \frac{2^{56}}{n \cdot L}$$

Donde:

- n: Número de procesos.

- L: Tamaño del lote de claves asignado a cada proceso en cada iteración.

Comparando esto con el enfoque "naive":

$$E[t_{Par}(n,k)]_{naive} = \frac{2^{55}}{n} + \frac{1}{2}$$

Se ve que el método DKSP tiene el potencial de ofrecer mejores tiempos paralelos esperados. Sin embargo, es importante considerar los costos de comunicación y gestión de lotes porque estos factores pueden afectar significativamente el rendimiento real del sistema.

3 Descripción del Acercamiento Propuesto: Parallel Random Search (PRS)

El "Parallel Random Search" (PRS) es un enfoque de búsqueda paralela en el que cada proceso genera y prueba claves de manera aleatoria. A diferencia de otros métodos que dividen el espacio de claves de forma estática o dinámica, PRS no asigna claves específicas a cada proceso. En cambio, cada proceso opera de forma independiente, generando claves aleatorias para realizar pruebas.

Algoritmo Descriptivo:

1. **Inicialización**: Cada proceso se inicializa con una semilla diferente para su generador de números aleatorios.
2. **Generación de Claves Aleatorias**: Cada proceso genera claves de manera aleatoria.
3. **Prueba de Claves**: Cada proceso prueba las claves generadas aleatoriamente para descifrar el texto cifrado y verifica si ha encontrado la clave correcta.
4. **Finalización**: Si un proceso encuentra la clave correcta, envía una señal a todos los demás procesos para que terminen. Si se agota el tiempo o se alcanza un número máximo de intentos, los procesos también pueden terminar.

4 Derivación del Valor Esperado de $t_{Par}(n,k)$ para PRS

Para el enfoque PRS, asumimos que cada proceso genera y prueba claves a la misma velocidad, y que la clave correcta puede estar en cualquier lugar del espacio de claves con igual probabilidad. Bajo estas condiciones, podemos aproximar el tiempo paralelo esperado $E[t_{Par}(n,k)]_{PRS}$ como:

$$E[t_{Par}(n,k)]_{PRS} \approx \frac{2^{55}}{n}$$

Esta fórmula es similar a la parte de generación de claves del enfoque "naive", pero no incluye el término adicional $\frac{1}{2}$ porque no hay un orden específico en el que las claves son probadas.

Al comparar PRS con los métodos "naive" y DKSP, PRS tiene un tiempo paralelo esperado más bajo que el método "naive"pero puede ser mayor que DKSP dependiendo del tamaño del lote L. Sin embargo, PRS tiene la ventaja de ser más sencillo de implementar y no requiere comunicación entre procesos para la distribución de claves por lotes.