Taxonomía de Duncan

Cristian Bladimir Mena Ochoa

Universidad Nacional del Altiplano

Facultad de Ingeniería Estadística e Informática

Docente: Fred Torres Cruz

Trabajo Encargado - N° 005

1 Taxonomía de Duncan

La Taxonomía de Duncan, presentada por Ralph Duncan en 1990, surgió tras un proceso de desarrollo entre los años 1988 y 1990, siendo publicada por primera vez en ese último año. En contraste con la taxonomía de Flynn, Duncan introdujo ajustes que permitieron la inclusión de procesadores vectoriales en cascada y otras arquitecturas que se consideraban aptas para ser clasificadas como arquitecturas paralelas, pero que no encajaban de manera fluida en el esquema original de Flynn. Este refinamiento se logró mediante la ampliación de los criterios de clasificación con subcategorías que representaban diferentes combinaciones de características arquitectónicas, abordando así aspectos de paralelismo a un nivel más detallado.

2 Arquitecturas Sincrónicas

Las arquitecturas síncronas coordinan operaciones concurrentes al mismo tiempo a través de relojes globales, unidades de control central o controladores de unidades vectoriales. Esta clase incluye subclases como: Procesadores Vectoriales, Matrices Sistolicas, y Arquitecturas SIMD. Las arquitecturas SIMD se clasifican además en matrices de procesadores y asociativas.

2.1 Procesadores Vectoriales

Los procesadores vectoriales se distinguen por su capacidad para ejecutar operaciones en paralelo sobre grandes conjuntos de datos. Hay dos tipos principales: los de registro a registro, donde los datos se manejan en registros de alta velocidad, y los de memoria a memoria, que utilizan búferes de memoria especializados. Estos procesadores, como el Cray-XMP/4 y el ETA-10, se han incorporado en supercomputadoras recientes, donde varios procesadores se conectan a través de una memoria compartida extensa. Aunque estas arquitecturas admiten paralelismo a nivel de tarea, destacan principalmente por su habilidad en el procesamiento vectorial.

- Memoria a memoria: CDC Star 100, Cyber 203, Cyber 205, TI-ASC
- Registro a registro: Cray-1, Cray-2, Cray-XMP/4, ETA-10, Fujitsu-200

2.2 Arreglos Sistolicas

Las Matrices de Procesadores Sistólicas son arreglos de procesadores diseñados para cálculos científicos intensivos, como procesamiento de imágenes y modelado nuclear. Surgieron en los años 60 y continúan evolucionando con sistemas como BSP. Estos procesadores manejan operandos de tamaño de palabra, generalmente valores de punto flotante de 32 a 64 bits. Una variante son los Procesadores Masivamente Paralelos (MPP), que consisten en muchos procesadores de un bit organizados en una cuadrícula. Ejemplos incluyen MPP de Loral, ICL DAP, y Modelos MasPar de MPP de Goodyear, entre otros.

2.3 Procesadores de memoria asociativa

Los procesadores de memoria asociativa se construyen alrededor de memorias asociativas y constituyen un tipo distintivo de arquitectura SIMD que utiliza una lógica de comparación especial para acceder en paralelo a los datos almacenados, de acuerdo con su contenido. La investigación en la construcción de memorias asociativas comenzó a finales de los años 1950, con el objetivo obvio de poder buscar en paralelo en las memorias aquellos datos que coincidieran con ciertos criterios especificados. La mayoría de los procesadores de memoria asociativa actuales utilizan la operación bit-slice (bit-columna), que implica operaciones concurrentes en un solo bit-slice (bit-columna) de

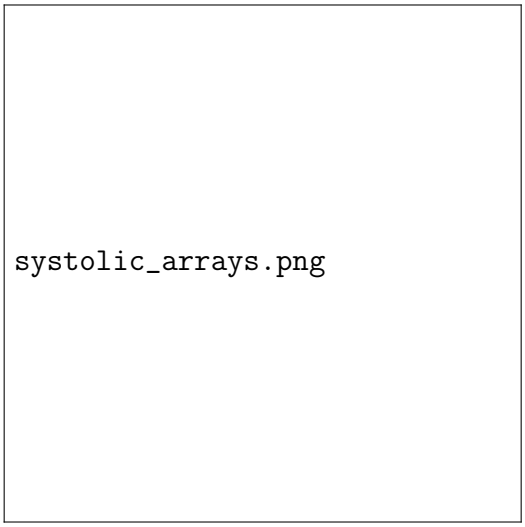

Figure 1: Arreglos Sistolicas

todas las palabras en la memoria asociativa. Ejemplos de computadoras de memoria asociativa son el Conjunto de Procesamiento de Elementos Paralelos (PEPE) de los Laboratorios Bell y el Procesador Asociativo (Aspro) de Loral.

3 Arquitecturas MIMD

Las arquitecturas MIMD emplean múltiples procesadores que pueden ejecutar instrucciones independientes en flujos de datos diferentes. Así, las computadoras MIMD admiten la ejecución paralela que requiere que los procesadores operen de manera mayormente autónoma. Aunque los procesos de software que se ejecutan en arquitecturas MIMD se sincronizan mediante el paso de mensajes a través de una red interconectada o accediendo a unidades de memoria compartida, las arquitecturas MIMD son computadoras asíncronas, caracterizadas por un control de hardware descentralizado. Por lo tanto, las arquitecturas MIMD también son conocidas popularmente como multiprocesadores y se dividen en dos subcategorías:

- Memoria compartida (acoplamiento estrecho).
- Memoria distribuida (acoplamiento flexible).

3.1 Multiprocesadores de memoria distribuida (acoplamiento flexible)

Las arquitecturas MIMD de acoplamiento flexible tienen memorias locales distribuidas conectadas a múltiples nodos de procesador. Las topologías de interconexión populares incluyen hipercubo, malla, mariposa y permutadores (refiriéndose a la Sección 1.5). El paso de mensajes es el método principal de comunicación entre los procesadores. La mayoría de los multiprocesadores están diseñados para ser escalables en rendimiento. Ejemplos de multiprocesadores acoplados de forma flexible son: DAMQ, Intel iPSC/2, nCUBE/10, Cube Serie 2010 de Ametek, Intel Personal Supercomputer, nCUBE/10, Supercomputadora Paralela Configurable de Alto Rendimiento (CHIP) de Lawrence Snyder y el Sistema SIMD/MIMD Particionable (Pasm) de Howard Siegel.

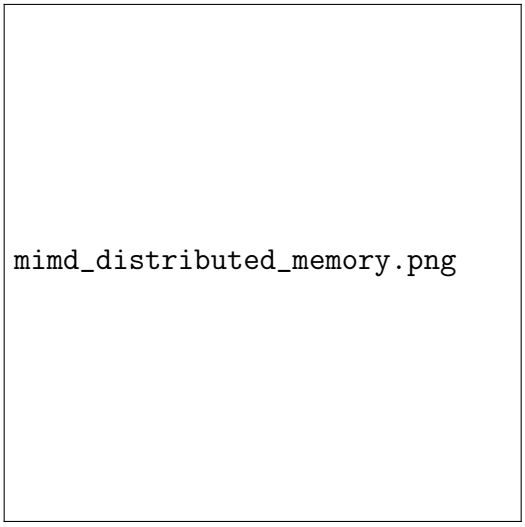

Figure 2: Arquitecturas MIMD-Memoria Distribuida

4 Paradigmas arquitectónicos basados en MIMD

Las diferentes paradigmas arquitectónicos no se ajustan fácilmente a la taxonomía tradicional de Flynn, como las arquitecturas híbridas MIMD/SIMD, de flujo de datos, de máquinas de reducción y procesadores de arreglos de

frente de onda. Aunque comparten características con el principio MIMD de ejecución y manipulación concurrente de flujos de instrucciones y datos, también tienen principios organizativos distintivos. Las arquitecturas MIMD/SIMD combinan porciones MIMD controladas por funciones SIMD, mientras que las de flujo de datos ejecutan instrucciones cuando sus operandos están disponibles, explotando paralelismo a diferentes niveles. El objetivo de estas arquitecturas es explorar nuevos modelos y lenguajes computacionales para computación de gran escala. Se mencionan algunos ejemplos como DADO, TRAC, Tagged Token DataFlow y LAU System.

5 Avances y Aplicaciones Modernas

Desde su propuesta, la taxonomía de Duncan ha evolucionado para incluir nuevas arquitecturas paralelas y ha demostrado ser útil en la clasificación de sistemas modernos. Por ejemplo, las arquitecturas de supercomputación actuales, como las utilizadas en el CERN y los laboratorios de investigación de clima, pueden ser clasificadas bajo las subcategorías de Duncan. Estas aplicaciones muestran la relevancia continua de la taxonomía en la investigación y desarrollo de sistemas paralelos de alto rendimiento.

6 Ventajas y Limitaciones

La taxonomía de Duncan ofrece varias ventajas, incluyendo una clasificación más detallada y precisa de las arquitecturas paralelas. Sin embargo, también enfrenta críticas por su complejidad y la dificultad de aplicarla a arquitecturas emergentes que combinan características de múltiples subcategorías.

6.1 Ventajas

- Clasificación detallada y precisa.
- Inclusión de arquitecturas vectoriales y paralelas modernas.

6.2 Limitaciones

- Complejidad en la aplicación.
- Dificultades con arquitecturas emergentes híbridas.

7 Ejemplos en Python y Comparación

7.1 Ejemplo SIMD

En este ejemplo, utilizamos numpy para realizar una operación de suma vectorizada, que representa una operación SIMD (Single Instruction, Multiple Data).

```
import numpy as np

# Datos
a = np.array([1, 2, 3, 4, 5])
b = np.array([5, 4, 3, 2, 1])

# Operación SIMD: Suma vectorizada
result = a + b
print("Resultado SIMD (Suma Vectorizada):", result)
```

Este ejemplo se clasifica como SIMD bajo la taxonomía de Flynn y como SIMD-Memoria Compartida en la taxonomía de Duncan.

7.2 Ejemplo MIMD

Para MIMD (Multiple Instruction, Multiple Data), usamos el módulo concurrent.futures para ejecutar operaciones independientes en paralelo.

```
from concurrent.futures import ThreadPoolExecutor

# Funciones que representan operaciones independientes
def add(x, y):
    return x + y

def multiply(x, y):
    return x * y

# Datos
values = [(1, 2), (3, 4), (5, 6), (7, 8), (9, 10)]

# Ejecutar operaciones en paralelo
```

```
with ThreadPoolExecutor() as executor:
    add_results = list(executor.map(lambda p: add(*p), values))
    multiply_results = list(executor.map(lambda p: multiply(*p), values))

print("Resultado MIMD (Suma):", add_results)
print("Resultado MIMD (Multiplicación):", multiply_results)
```

Este ejemplo se clasifica como MIMD bajo la taxonomía de Flynn y como MIMD-Memoria Distribuida en la taxonomía de Duncan.

8 Referencias

- C. Xavier and S. S.lyengar. *Introduction to parallel algorithms*. En: Wiley. United States of America; 1998. p. 25-29.
- I.M. Willers and Cern-Cn. *Parallel Computers*. Academic Training Programme, 1990.
- Andre Przybysz. *Classificação de arquitetura de processadores Duncan x Flynn*. ANDRÉ PRZYBYSZ, 2023.