وزارة التعليم العالي و البحث العلمي

جامعة الجزائر 3

كلية العلوم الاقتصادية و العلوم التجارية و علوم التسيير

قسم العلوم الاقتصادية

أطروحة مقدمة لنيل شهادة دكتوراه علوم في العلوم الاقتصادية

تخصص : **الاقتصاد القياسي**

الخوارزميات المتوازية و تطبيقاتها في بحوث العمليات

اعداد الطالب : **اشراف الاستاذ:**

رياش لخضر **خليد علي**

لجنة المناقشة:

أ. غريس عبد النور........ جامعة الجزائر 3 .... رئيسا

أ. خليد علي................ جامعة الجزائر 3...........................مقررا

أ . بوزارة العيد .... جامعة الجزائر 3......................... ممتحنا

أ. حديد نوفل ....المدرسة الوطنية التحضيرية لدراسات مهندس....ممتحنا

أ. طويبة أحمد ....المدرسة العليا للإحصاء................. ممتحنا

أ. علوطي لمين .... جامعة المدية ....ممتحنا

السنة الجامعية: 2018/2017

شكر و تقدير

أشكر الله سبحانه و تعالى الذي ألهمني الطموح و سدد خطاي .

و أتقدم بجزيل الشكر و العرفان للأستاذ الدكتور خليد علي الذي أشرف على هذا العمل و لم يبخل بجهد أو نصيحة وكان مثالا للأستاذ المتواضع.

كما أشكر الاساتذة الكرام أعضاء لجنة المناقشة على تفضلهم بقبول مناقشة هذه الاطروحة .

لا يفوتني أن أخص بالشكر الجميل إلى زوجتي العزيزة على مساندتها لي و أخي عبد الحميد و مبروك .

والشكر موصول إلى الأستاذ قبايلي على مساعدته لي .

فى الاخير أشكر كل من ساعدني من قريب أو من بعيد في انجاز هذا العمل .

لخضر رياش

الإهداء

أهدي هذا العمل الى :

كل العائلة الكريمة

كل أساتذتي في جميع أطواري الدراسية

فهرس المحتويات

فهرس الأشكال

قائمةالجدوال

المقدمة العامة

مقدمة:

في النصف الثاني من القرن العشرين ولد علم المعلوماتية وقد تطور هذا العلم بشكل كبير وبسرعة مذهلة وتشعبت فروعه وتعددت مجالاته. حيث واكب هذا العلم كافة العلوم الأخرى. كما قادت ضرورة تصميم الخوارزميات الملائمة لحل المسائل المطروحة إلى تطور كبير في علم الخوارزميات. واقتضت بنية الحاسوب التي رافق ظهورها ولادة علم المعلوماتية إلى وضع خوارزميات تنفذ بالتتابع بحيث تكون تعليمات البرنامج مرتبة ترتيبا دقيقا وتنفذ الواحدة تلوى ألأخرى من بداية حتى نهاية البرنامج. إلا أنه ومع زيادة حجم المسائل الواجب حلها وتشعبها وصعوباتها أصبح من الصعب على مثل هذه الخوارزميات الوصول لحل دقيق لتلك المسائل إما لتجاوز قدرات الذاكرات اللازمة لتخزين المعطيات والنتائج المرحلية أو بسبب المدة الزمنية الكبيرة اللازمة للوصول للحل ، مما يؤدي الى توقف الحواسيب قبل التوصل للحل الأمثل ، الأمر الذي أدى بالعلماء والباحثين للتفكير في ايجاد أساليب جديدة قد تساعد على تسريع عملية البحث عن الحلول المطروحة أو زيادة سعة و قدرات تلك الذاكرات.

" و من هنا ولدت مسألة التوازي في البرمجة واستخدام الشبكات الموزعة إلا أنه تبين استحالة تطبيقها على الأجهزة التقليدية التي تعتمد الالكترونية الثابتة ، الأمر الذي استدعى التدخل في بنية الجهاز نفسه والبحث عن بنية جديدة غير بنية فون نيومان التقليدية. وتم في مطلع الثمانينات تصميم أول حاسوب متعدد المعالجات سمي CRAY1 الذي احتوى على معالجين يعملان في آن واحد ، وتوالى بعد ذلك ظهور حواسيب جديدة تعتمد على بنى مختلفة. وكان القاسم المشترك بين مختلف هذه البنى هو استخدام عدة معالجات في الحاسوب الواحد ، و على الرغم من الامكانيات الهائلة التي تقدمها هذه الأجهزة سواء في قدرتها الكبيرة في الحساب أو في حجم الذاكرات الكبير جدا ، إلا أن ظهورها طرح مشكلة أساسية هي عدم المحافظة على التسلسل الزمني لتنفيذ التعليمات والعمليات مما استدعى تطوير خوارزميات ، فظهرت الخوارزميات المتوازية لتتلاءم مع العتاد الجديد الذي فرضته هذه الحواسيب[1]" .

تعتبر مسائل الأمثلية في بحوث العمليات جزء من حياتنا اليومية و تشمل العديد من المهام كحساب أقصر مسار(مثل حساب المسارات باستعمال برامج GPS) أو حقيبة الظهر(تعظيم قيمة الأغراض الواجب حملها من طرف المسافر) أو مسألة التعيين (مثل توزيع محلات التخزين) أو كذلك إيجاد شجرة بأدنى تغطية(مثل شبكة المياه ، شبكة الهاتف ، شبكة الكهرباء ، ...). حجم هذه المسائل في

[1] :كندة زين العابدين ، "خوارزميات المعالجة المتوازية وبرمجتها"، جامعة دمشق، ص1

ارتفاع مستمر و قوة الحساب الضرورية لمعالجتها كذلك ، مما أدى بنا إلى التفكير في استعمال الخوارزميات المتوازية قصد الوصول للحل بسرعة.

في هذا السياق يمكن صياغة الاشكالية التالية :

ما مدى فعالية الخوارزميات المتوازية في حل بعض مسائل بحوث العماليات عوض الطرق التسلسلية؟

من خلال دراستنا لهذا الموضوع نحاول الاجابة على مجموعة من الاسئلة الفرعية التالية :

- ما هي الخوارزميات المتوازية ؟ و ما هي خصائصهما؟
- ما هي أهم خصائص الحاسبات التي يمكن أن تنفذ عليها؟
- كيف يمكن تصميم هذه الخوارزميات؟
- كيف يمكن دراسة و تحليل خصائصها؟
- كيف يمكن مقارنتها بالخوارزميات التقليدية المتسلسلة؟
- ما هي المزايا المنتظرة من هذه الخوارزميات المتوازية وخاصة في ميدان بحوث العمليات؟

فرضيات البحث:للإجابة على هذه الإشكالية تمت صياغة بعض الفرضيات:

- الخوارزميات المتوازية تكون أسرع تنفيذا من الخوارزميات المتسلسلة كلما زاد عدد المعالجات.
- الخوارزميات المتوازية تكون أسرع تنفيذا من الخوارزميات المتسلسلة كلما زاد حجم البيانات في الذاكرة.

أهمية الدراسة : تتمثل أهمية هذا الموضوع في التقدم الحاصل حاليا في استخدام علم المعلوماتية بوصفه الأداة المفضلة للمساعدة على حل المسائل الصعبة في العديد من المجالات ومحاولة تطبيقه في حل مسائل بحوث العمليات.

لقد تم اختيار علم بحوث العمليات لتطبيق هذه الخوارزميات نظرا لكثرة المسائل التي يعالجها هذا العلم من جهة ، ولوجود مسائل قياسية تم حلها بشكل نهائي بالأساليب التسلسلية ، مما يسمح لنا بإجراء مقارنة بين هذه الخوارزميات و تحديد و حساب بعض المعايير للقيام بذلك. نذكر من بين هذه المعايير مثلا: حساب زمن التنفيذ ، حساب التسريع و حساب الفعالية .

أهداف الدراسة : تكمن الأهداف الرئيسية لهذا العمل في:

أ- عرض مفاهيم المعالجة المتوازية في الحاسوب ، أنواعها وخصائصها.

ب- التعريف بمبادئ تصميم خوارزميات متوازية أو تكييفها في حالات الحساب المتوازي على الأجهزة متعددة المعالجات.

ج- تطبيق التوازي في خوارزميات الترتيب و خوارزميات بحوث العمليات و مقارنتها بصيغها التسلسلية.

د- محاولة كتابة أمثلة لبرامج في لغة البرمجة المتوازية Cuda.

منهج الدراسة : للإجابة على إشكالية البحث و تحقيق أهدافه ، نستخدم في الدراسة المنهجين الوصفي و التحليلي. ويتم الاعتماد على الأسلوب الوصفي للإطار النظري للتعريف بالخوارزميات المتوازية. أما الجانب التحليلي فيظهر من خلال تطبيق بعض هذه الخوارزميات في ترتيب المعطيات و في بحوث العمليات .

خطة البحث : بغية تحقيق هدف الدراسة و اختبار الفرضيات قسمنا البحث إلى أربعة فصول.

الفصل الأول : طرق المعالجة في الحاسوب ، خصص إلى عرض مفاهيم عامة حول طرق المعالجة (التسلسلية ، المتوازية ، الموزعة) في الحاسوب وخصائصها موضحين ذلك من خلال أشكال و أمثلة، بعد ذلك تطرقنا إلى أهمية الحاجة لاستخدام التوازي أو التوزيع في البرمجيات أو العتاد وما له من فوائد في فعالية و سرعة الأداء للحاسوب. كما تناولنا مختلف أشكال معالجة المتوازية للمعطيات على مستوى كل من البرنامج ، الإجرائية والتعليمة و في الأخير، تطرقنا لتاريخ الحاسبات و مسايرة الخوارزميات لها عبر مختلف الأجيال.

الفصل الثاني: تصنيف الحاسبات والخوارزميات المتوازية ، خصص هذا الفصل إلى تصنيف فلاين (Flynn's classification) للحاسبات المتوازية حسب توازي المعطيات أو توازي التعليمات أو بعبارة أخرى حسب كمية تدفق المعطيات و كذلك حسب كمية تدفق التعليمات وهي:

أ- حاسبات وحيدة تدفق التعليمات و وحيدة تدفق المعطيات (SISD).

ب- حاسبات وحيدة تدفق التعليمات ومتعددة تدفق المعطيات (SIMD).

ج- حاسبات متعددة تدفق التعليمات ووحيدة تدفق المعطيات (MISD).

د- حاسبات متعددة تدفق التعليمات ومتعددة تدفق المعطيات (MIMD).

موضحين ذلك بأشكال و أمثلة لكل صنف ، ثم تناولنا أنواع شبكة الربط بين المعالجات سواء منها السكونية أو الديناميكية وخصائص كل منها. بعد ذلك قمنا بتعريف مختلف نماذج الخوارزميات المتوازية مثل نموذج توازي المعطيات ، نموذج الرسم البياني للمهام ، نموذج السيد و العبد و أخيرا النموذج الهجين. ووضحنا كل هذه النماذج بأشكال.

الفصل الثالث: تصميم الخوارزميات المتوازية ، خصص لبعض المفاهيم الأساسية و المستعملة في تصميم الخوارزميات المتوازية مثل:

التقسيم ، المهام ، التزامن ، مخطط التبعية ، درجة التزامن ، التفاعل بين المهام. تطرقنا أيضا إلى مختف تقنيات التقسيم كالتقسيم العَوْدِي ، تقسيم البيانات ، التقسيم الاستكشافي و التقسيم التخميني ، مع إعطاء أمثلة وخوارزميات توضيحية لكل أسلوب على غرار خوارزمي الترتيب الفردي-الزوجي و خوارزمي الفرز السريع في صيغها التسلسلية و المتوازية و أمثلة توضيحية.ثم بعد ذلك عرضنا مختلف الأساليب العملية الأساسية للتوازي في الخوارزمي و وضحنا كل منها بأمثلة.

الفصل الرابع : تطبيقات الخوارزميات المتوازية في بحوث العمليات ، تطرقنا في الفصل الرابع إلى إعطاء تعريف بعض المفاهيم في بحوث العمليات وخاصة منها تلك المتعلقة بنظرية البيان. سنقوم بعد ذلك بإعطاء أمثلة لخوارزميات في نظرية البيان في صيغتها التسلسلية وهي معروفة كخوارزمي Prim (Prim's algorithm) وكريسكال (Kruskal) لإيجاد الشجرة بأقل تغطية وكذلك خوارزمي Dijkstra و خوارزمي Floyd لإيجاد أقصر مسار ثم محاولة ايجاد صيغته المتوازية و مقارنتها بصيغته المتسلسلة بالاعتماد على حساب معيار التعقيد الزمني ، التسريع والفعالية. وقد اخترت هذين المثالين لأهميتهما التطبيقية في الاقتصاد(شبكة الهاتف ، شبكة الغاز ، شبكة الكهرباء ، ربط الطرقات...) وهذا طبعا بأقل تكلفة.

بعد ذلك ، سنقوم بوصف لبعض لغات البرمجة المتوازية منها على سبيل المثال:

CSP(Communicating Sequentiel Process), OCCAM ,OpenMP و Pthreads ,Parallel Fortran, Parallel Pascal, Parallel C,CUDA : Compute Unified Device Architecture) ,MPI(Message Passage Processing)

ثم سنحاول إعطاء بعض البرامج كملحق في لغة البرمجة التسلسلية C و المتوازية كودا (CUDA).

في الأخير خاتمة عامة ألخص فيها أهم النتائج المتوصل إليها و أهم التوصيات التي نرجوها أن تكون في الدراسات المستقبلية.

دراسات سابقة : تطرقت العديد من الدراسات إلى مسألة الخوارزميات المتوازية و الفائدة من التوازي مقارنة بالخوارزميات المتسلسلة في مختلف الميادين التطبيقية.

نقوم في هذه الأطروحة بعرض بعض الدراسات التي اختارت كتطبيق ميدان بحوث العمليات.

دراسة بعنوان " Le calcul multi GPU et optimisation combinatoire" لمؤلفهاَ Abdelhamid Boukedar لسنة 2010-2011.

الهدف من الدراسة هو وضع نظام CPU/GPU لمختلف خوارزميات مسائل الأمثلية من نوع حقيبة الظهر، و بالأخص خوارزميات Branch and Bound ثم بعد ذلك دراسة وتحليل مدى نجاعة هذه الطرق و المسماة بالهجينة (hybrid) أي استعمال المعالج الرئيسي (CPU) ومعالج الرسومات (GPU) للحاسوب في عمليات الحساب.

نتائج الدراسة:

- مقارنة زمن التنفيذ الخوارزمي بالثانية بين المعالج CPU و المعالج GPU مع تغيير عدد الوضعيات.

عدد الوضعيات (etat) = 1000

عدد المتغيرات	زمن الحساب GPU(s)	زمن الحساب CPU(s)
100000	**1.36**	**1.18**

عدد الوضعيات (etat) = 10000

عدد المتغيرات	زمن الحساب GPU(s)	زمن الحساب CPU(s)
100000	1.37	1.78

عدد الوضعيات (etat) = **100000**

عدد المتغيرات	زمن الحساب GPU(s)	زمن الحساب CPU(s)
100000	**2.18**	**17.78**

نلاحظ من خلال هذه الجداول أن زمن الحساب (زمن التنفيذ) GPU لحل الخوارزمي يكون أقل بكثير من زمن الحساب CPU و هذا كلما أرتفع عدد الوضعيات و خاصة إذا تجاوز عدد الوضعيات 100000.

نشير إلى أن عدد خيوط التنفيذ أثناء تطبيق الخوارزمي في كل الحالات كان:1024 thread/block .

- مقارنة زمن التنفيذ المعالج GPU مع تغيير عدد Thread/Block وهذا ل100000 متغيرة.

عدد خيوط التنفيذ(thread/block)	زمن الحساب GPU(s)
1024	2.18
512	1.87
256	**1.87**
192	1.88

يتبين من الجدول أن عدد thread/Block له كذلك دور مهم في خفض زمن الحساب GPU.

خلصت الدراسة في الأخير إلى ما يلي:

بالرغم من التقدم التكنولوجي وسرعة التنفيذ للمعالجات (CPU). هذه الأخيرة لم تتمكن بعد من تخفيض زمن التنفيذ لإيجاد الحل الأمثل للمسائل المصنفة بالصعبة. هناك حل أقترح باستعمال البطاقات الرسومية GPGPU ل Nvidea و الذي يمكنه اعطاء تسريع (speedup) أفضل من المعالج (CPU).

دراسة تحت عنوان **standard « Etude et implementation de l'algorithme de simplexe sur GPU »**

لصاحبها Xavier Meyer من كلية المعلوماتية جامعة جنيف سنة 2011.

هدف الدراسة هو تحليل و برمجة خوارزمي السمبليكس (simplex) على المعالج GPU. هذا الخوارزمي له العديد من البرامج في صيغها المتسلسلة ، ولكن أغلبها لها نجاعة جيدة للمسائل الصغيرة أو متوسطة الحجم فقط. ولهذا كان الاهتمام بطريقة التوازي من خلال GPGPU. (General-Purpose Computing on Graphics Processing Units)

تبدأ الدراسة بمقارنة طريقتي السيمبلكس (standard et révisée) ثم اختيار الطرقة المناسبة لمعالجتها على المعالج GPU. و قد تم أختيار طريقة السيمبلكس standard والتي تبدو أكثر ملائمة للمعالجة المتوازية على المعالج GPU. لقياس النجاعة لكل برنامج كان لا بد من اختيار اصدار تسلسلي موجود ومرجعي لسيمبلكس قصد مقارنة نتائجها. وقد أظهرت هذه المقارنة زيادة في النجاعة و هذا خاصة للمسائل متوسطة وكبيرة الحجم.

دراسة بعنوان"**الطرائق التكرارية المتوازية**" لمؤلفها **محمد واجد محمد علي** من جامعة الموصل و هي عبارة عن بحث يهدف إلى تطوير طرائق متوازية للطريقتين التكراريتين جاكوبي و كاوس_سيدل و اللتان تستعملان في البرمجة الخطية لحل منظومات من النماذج الخطية.
وكما هو معروف فإن هذه النماذج تتطلب وقتا طويلا للتنفيذ عند المعالجة باستخدام المعالج التتابعي. يحاول الباحث تقليل هذا الوقت و زيادة كفاءة الخوارزميات للطريقتين من خلال اقتراح طرائق متوازية ملائمة على حاسبات من نوع MIMD.
و في الأخير، توصل الباحث إلى النتائج و التي لخصها في الجداول التالية:
جدول (1) : يمثل الوقت المحسوب بالثانية في تنفيذ طريقة جاكوبي و كاوس-سيدل التكرارية مع الطرائق المتوازية ولعدد من التكرارات (20) تكرار.

سعة المصفوفة	طريقة جاكوبي التكرارية	طريقة جاكوبي المتوازية الأولى	طريقة جاكوبي المتوازية الثانية	طريقة كاوس–سيدل التكرارية	طريقة كاوس سيدل المتوازية
50x50	2,39 (1 CPU)	1,17 (3 CPU)	0,764 (51 CPU)	1,68 (1 CPU)	0,8757 (3 CPU)
100x100	9,48 (1 CPU)	4,79 (3 CPU)	3,142 (101 CPU)	7,024 (1 CPU)	3,522 (3 CPU)
200x200	37,6 (1 CPU)	19 (3 CPU)	10,28 (201 CPU)	28,138 (1 CPU)	14,241 (3 CPU)

ملاحظة : برمجت هذه الطرائق المتوازية الثلاث و تنفيذها باستخدام لغة البرمجة ++c على حاسب PIV بمعالج تردده 2GHz و باستخدام دالة(Delay120000) لعدة أنظمة ، و ثم تنفيذ كل اجراء على حدا و حساب الوقت له لأن حاسبات MIMD غير متوفرة في بلاده حسب الباحث.

جدول(2) : يمثل عامل التسريع للطرائق المتوازية

سعة المصفوفة	طريقة جاكوبي المتوازية الأولى	طريقة جاكوبي المتوازية الثانية	طريقة كاوس–سيدل المتوازية
50x50	2,3751066	3,1286722	1,91846523
100x100	1,97994987	3,1718651	1,99432141
200x200	1,98469657	3,6575875	1,97584439

توصل الباحث و من خلال ملاحظة عامل التسريع في الجدول(2) أن عامل تسريع الطريقة الثانية أقل من الطريقتين الاولى والثالثة و أن عامل التسريع في الطريقة الثالثة هو أفضل من الطريقة الاولى، اضافة الى أن عامل التسريع في الطريقتين الاولى و الثالثة يزداد بنسبة ملحوظة عن الطريقة الثانية إذا ازدادت سعة المصفوفة, وتوصل أيضا من خلال مقارنة بين الطرائق المتوازية المقترحة إلى نتائج لخصها في الجدول(3),

جدول(3) : الفروق بين الطرائق المتوازية المقترحة لطريقتي جاكوبي و كاوس–سيدل التكراريتين,

طريقة جاكوبي المتوازية الأولى	طريقة جاكوبي المتوازية الثانية	طريقة كاوس–سيدل المتوازية
نحتاج إلى 3من المعالجات مهما كان عدد المعادلات	نحتاج ل n+1 من المعالجات إذ أن n هو عدد المعادلات	نحتاج إلى 3 معالجات مهما كان عدد المعادلات
عندما تكون المسألة كبيرة تأخذ وقتا كبيرا	عندما تكون المسألة كبيرة يكون وقت التنفيذ أقل من الطريقة الأولى,	عندما تكون المسألة كبيرة تأخذ وقتا كبيرا
غير مكلفة بالنسبة للمسائل الكبيرة لأن تحتاج إلى 3 معالجات	تعد مكلفة للمسائل الكبيرة, لذا يفضل استعمالها في المسائل الصغيرة	غير مكلفة بالنسبة للمسائل الكبيرة لأ كافة المسائل تحتاج إلى 3 معالجات,
الحمل يكون على المعالجات بشكل متساو تقريبا	الحمل على CPU1 و CPUn متساو و الحمل يكون أكثر على بقية المعالجات,	الحمل يكون على CPU1 و CPU2 بشكل متساو تقريبا، و الحمل الأكبر يكون علىCPU3

دراسة بعنوان "**الحاسبات المتوازية و الخوارزميات المتوازية** " قامت بها مجموعة من الباحثين بإشراف الباحث عماد فتاش من كلية العلوم ، قسم علوم الحاسب الآلي ، جامعة الملك سعود.

تعرضت الدراسة إلى التعريف بمفاهيم حول المعالجة المتوازية من حيث العتاد و البرمجيات. ففي جزئها الأول تناولت مفهوم التوازي ، مفهوم الحاسب المتوازي ، الحاجة للتوازي إلى غير ذلك.

في الجزء الثاني منها، تطرقت إلى تصنيف فلاين للحاسبات حسب تدفق المعطيات أو تدفق التعليمات و هي حاسبات SISD ، حاسبات SIMD ، حاسبات MISD و حاسبات MIMD و كذلك إلى مختلف أنواع شبكات الربط بين معالجات الحاسب المتوازي. منها على سبيل المثال: شبكات خطية ، حلقية ، نجمية ، شجرية ، مكعبيه إلى غير ذلك. و قد وضحت بأشكال و أمثلة وافية. في الجزء الثالث، تناولت الدراسة مفاهيم تدخل في تصميم الخوارزميات المتوازية نذكر مثلا تقنيات تقسيم العمليات الحسابية و أنواعها، المهام ، مخطط التبعية بين المهام ، الحبوبية (أي حجم المهمة)، المقابلة ، التفاعل بين المهام موضحا ذلك بأمثلة للخوارزميات الفرز أو البحث عن أصغر قيمة في قائمة أو ضرب مصفوفات...

في الجزء الرابع و الأخير، تناولت الدراسة بعض لغات البرمجة المتوازية و خصائص كل منها.

تظهر هذه الدراسات السالفة الذكر أن زمن تنفيذ خوارزمياتها في صيغتها المتوازية أقل بكثير منها عندما تنفذ بالتتابع على معالج وحيد و هذا واضح خاصة في الدراسة الأولى و الثانية و الثالثة. أما الدراسة الرابعة ففي الجانب النظري للمعالجة المتوازية كانت وافية و ما ينقصها في نظري هو الجانب التطبيقي و هو ما حاولت في أطروحتي العمل عليه من خلال تحليل الخوارزميات المتوازية و مقارنتها بصيغها التسلسلية و هذا بالاعتماد على حساب معايير كالتعقيد الزمني (زمن التنفيذ النظري)، التسريع و الفعالية ثم استنتاج فعاليتها من عدمها.

الفصل الأول

طرق المعالجة في الحاسوب

تمهيد : كانت العمليات الحسابية في السنوات الماضية في الحاسوب تتم بالتتابع من خلال تنفيذ برنامج متكون من عدة تعليمات و معطيات تكون كلاهما مخزنة في الذاكرة الحية من طرف المعالج و هذا من البداية حتى النهاية. و مع مرور الوقت ظهر الحساب المتوازي والحساب الموزع وكانا مقتصران في البداية في معامل الأبحاث ، و لكن اليوم فهما متوفران و على نطاق واسع في السوق. إن مجال المعالجة المتوازية و الموزعة قد تطور و أصبح يدرس في المراحل الجامعية. كما نلاحظ أن التوازي أصبح يستعمل بشكل واسح في ميدان الحاسبات سواء في تصميم مكونات العتاد إلى غاية تصميم البرمجيات المتوازية و المعقدة . و لقد أصبحت المعالجة المتوازية في وقتنا هذا في مقررات علوم الحاسوب ، مثل دراسة بنية الحاسبات ، أو البرمجة ، أو الشبكات أو الخوارزميات وغيرها. و نفس الشيء يمكن قوله على الحساب الموزع..

1- مفاهيم عامة:

نتطرق في ما يلي لأهم مفاهيم و مبادئ المعالجة في الحاسوب.

1-1 :أنواع المعالجة في الحاسوب:

هناك أنواع مختلفة للمعالجة في الحاسوب نذكر مثلا:

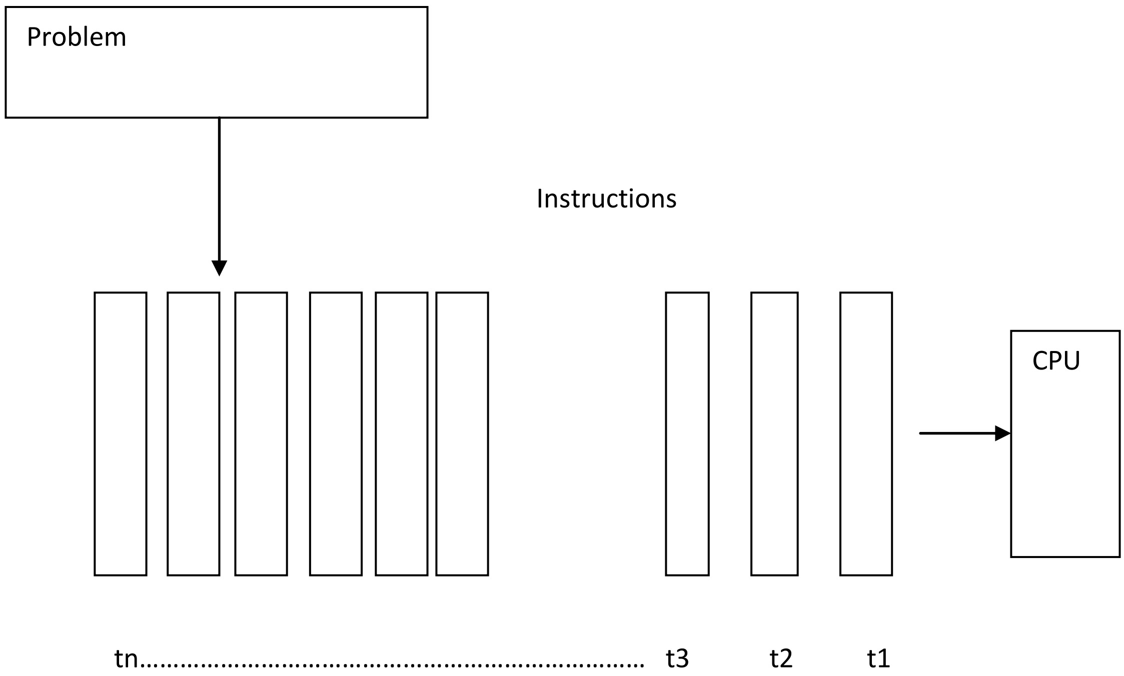
أ- **الحساب المتسلسل (sequential computing)**: هو الحساب الذي يمكن تنفيذه على حاسوب واحد مكون من وحدة مركزية واحدة أي معالج (CPU).

- تقسم المسألة إلى سلسلة من التعليمات
- تنفذ التعليمات الواحدة تلوى الأخرى
- تنفذ تعليمة واحدة فقط في كل زمن معين t.

ويمكن تمثيلها وفق الشكل التالي [1] :

[1] : Amara Yacine , Le GPU : un moyen pour le calcul intensif , Ecole Militaire Polytechnique , Alger ,P4

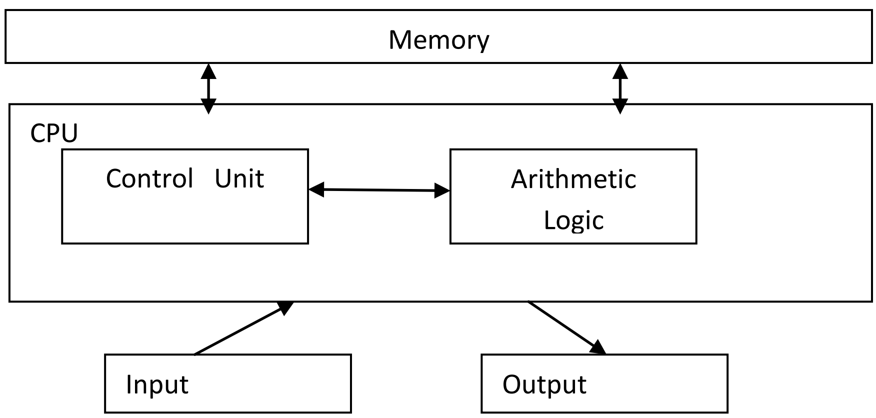
الشكل(1-1) : الحساب المتسلسل.



مصدر: Amara Yacine , Le GPU : un moyen pour le calcul intensif , Ecole Militaire Polytechnique , Alger,p.4

أ- بنية (Von Neumann) للحاسوب: بنية **Von Neumann** ، هو نموذج لحاسوب متسلسل يستعمل بنية تخزين وحيدة لحفظ التعليمات والمعطيات المطلوبة أو الناتجة من الحساب[2]. و يمكن تمثيله في الشكل(1-2) التالي:

الشكل (1-2):حاسوب متسلسل عمارة فون نيومان (Von Neumann)



المصدر: Amara Yacine ، مرجع سبق ذكره، ص.5

[2] : Amara Yacine, p.5

في هذه الحالة الحاسوب مقسم إلى أربع أجزاء رئيسية:

الوحدة الحسابية والمنطقية(Arithmetic Logic Unit): يتمثل دورها في إجراء العمليات الأساسية الحسابية منها والمنطقية (الجمع، الضرب،........).

وحدة التحكم أو المراقبة (Control Unit): يتمثل دورها في ترتيب العمليات.

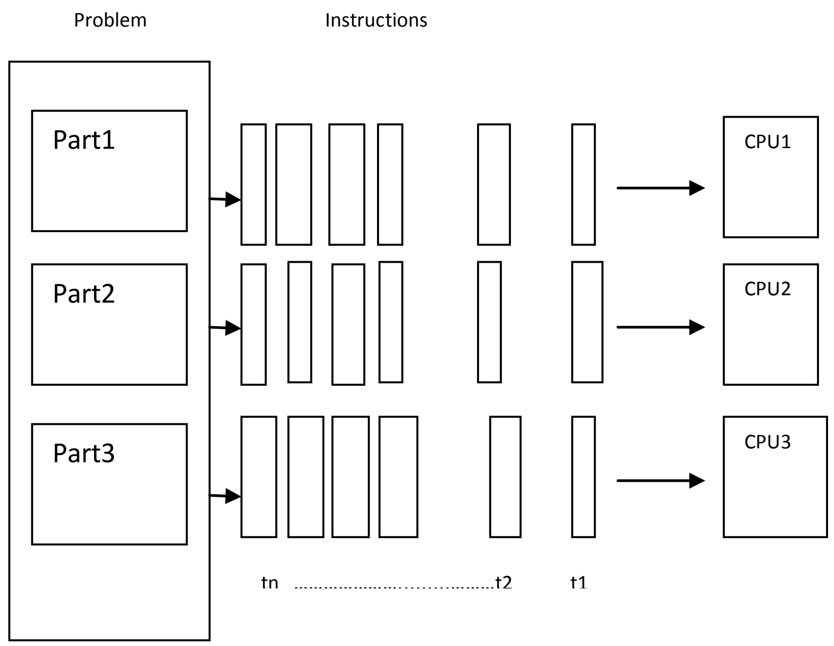
الذاكرة الأساسية(Memory): تحتوي في نفس الوقت على المعطيات والبرنامج الذي يبين لوحدة المراقبة ما هي الحسابات المطلوبة على هذه المعطيات.

وحدات الإدخال و الإخراج(Input/Output): والتي تسمح بالاتصال بالمحيط الخارجي.

ب- الحساب المتوازي (parallel computing): الحساب المتوازي هو الاستعمال لعديد موارد الحساب لحل مسألة [3].

- المسألة مقسمة إلى أجزاء منفصلة والتي يمكن حلها في نفس الوقت.
- كل جزء مقسم هو كذلك إلى سلسلة من التعليمات.
- تعليمات كل جزء يمكن تنفيذها في نفس الوقت و على معالجات مختلفة .

الشكل (1-3): الحساب المتوازي



المصدر: Amara Yacine ، مرجع سبق ذكره، ص12

---

[3]: **Amara Yacine ، مصدر سبق ذكره ، ص12**

هذه الموارد قد تكون حاسوب واحد متعدد المعالجات، عدد معين من الحواسب مرتبطة معا،أو الاثنان معا.

لا يقتصر مفهوم التوازي على القيام بالحساب في الحاسوب و لكن هو حقيقة موجودة و ممارسة في مختلف أنشطتنا في الحياة اليومية. فعلى سبيل المثال سماع الموسيقى أثناء مراجعة الدروس تعتبر أنشطة متوازية. و عليه فمفهوم التوازي أوسع من أن يكون محصورا في ميدان الحساب و بالتالي يمكن القول أن **التوازي** (**parallelism**) : "**هو مجموعة من الأنشطة التي تحدث في نفس الوقت**".[4]

و يمكن أن نفرق بين مفهوم التزامن (Concurrency) و مفهوم التوازي.

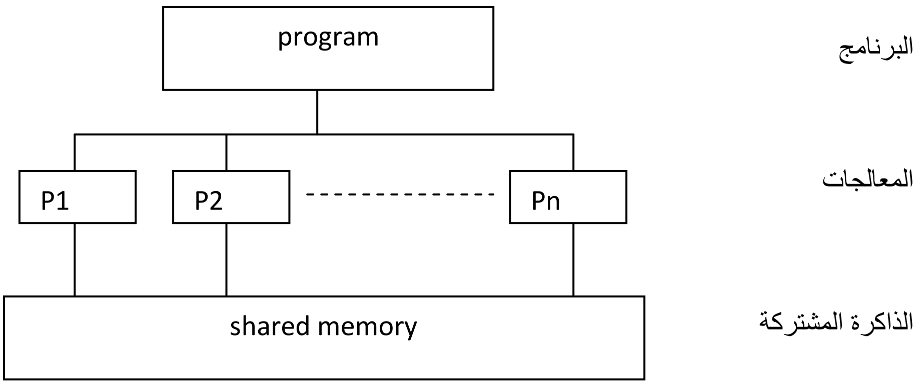
فالتزامن(Concurrency) : "**هو القدرة على التشغيل في نفس الوقت**".[5] يجب أن نفرق بين التعريفين السابقين.. فالتوازي يستخدم للإشارة إلى الأنشطة التي تحدث في نفس الوقت ، على سبيل المثال أربع مهام يتم تنفيذها على أربع معالجات (CPUs) في نفس الوقت. أما التزامن فهو أوسع .فيشير إلى كلا الحالتين الأولى التي تعتبر متوازية حقا و الأخرى التي فيها " توازي وهمي "، و كمثال للتزامن: أربع مهام يقسم بينها الوقت و تنفذ على معالج واحد (Time sharing system) [6] .

ج- الحساب الموزع (distributed computing): الحساب الموزع هو حساب يجرى على جهاز متوازي متكون من عدة وحدات حساب، كل واحدة مستقلة عن الأخرى تكون المسافة بينها معتبرة جدا، تتطلب بالتالي استعمال شبكة تواصل واسعة(مثل الأنترنت)[7] . إن الهدف الأول من استعمال الأنظمة الموزعة هو لزيادة قدرة الحساب (بتكلفة أقل) ويتم هذا عن طريق الزيادة التدريجية لوحدات الحساب أو بتجميع المجموعات الموجودة. فعلى سبيل المثال النظام GRID5000 متكون من 5000 حاسوب موزع على كل التراب الفرنسي. أما الهدف الثاني فيتمثل في تقاسم بنوك المعطيات الموزعة بينها، و يتم التواصل بين أجهزه النظام عبر مختلف شبكات الأتصال(أنترنت، أنترانت، الإكسترانت، و الشبكات الإجتماعية للمؤسسات......).

[4] :DANIEL C. Hyde, "Introduction to the Principles of Parallel Computation", Bucknell University, 1998, p7
[5]: DANIEL C. Hyde ، مصدر سبق ذكره ، ص7
[6] : DANIEL C. Hyde ، مصدر سبق ذكره ، ص7
[7] : Cyrile Gavoile, " Algorithmes distribués , Université de bordeaux" , (2005), P5

- **الإطار القاعدي للأنظمة الموزعة:** و يقصد به الوحدات المكونة للنظام قد تكون معالجات، حواسيب، ساتل. كما يوجد نموذجان أساسيان للبنية أو العمارة الموزعة "تبادل الرسائل" " والذاكرة المشتركة" :
  - نموذج تبادل الرسائل: يعالج هذا النموذج الاتصالات بتبادل الرسائل. يعني عندما يريد معالج التواصل مع آخر، يجب أن يرسل له رسالة عبر الوسيط.
  - نموذج الذاكرة المشتركة: في هذا النموذج المعالجات لا تتواصل فيما بينها مباشرة بعكس النموذج السابق، وإنما تتواصل عبر منطقة من الذاكرة مشتركة في القراءة و/أو في الكتابة. أي الذاكرة المشتركة يمكن استعمالها للتواصل[8] . و يعتبر نموذج PRAM حالة خاصة لنموذج الذاكرة المشتركة و الشكل(4-1) يوضح ذلك.

الشكل (4-1): النموذج (PRAM : Parallel Random Acces Machine)



المصدر: Cyril Gavoile , Algorithmes distribués , Université de bordeaux , (2016), P.10

يتميز الحساب الموزع بعدة خصائص نذكرمن بينها :

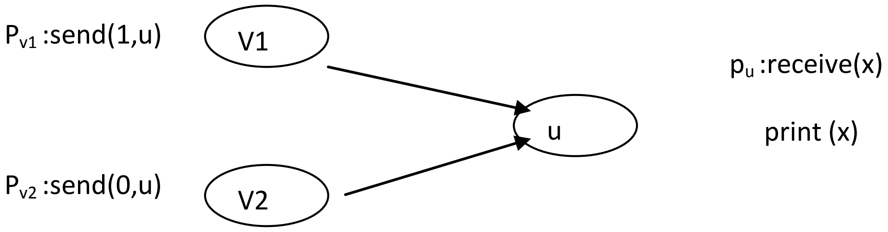
- التواصل بينها ليس مجاني، في أغلب الأحيان يكون زمن الحساب أقل بكثير من زمن التواصل.
- المعالج له إدراك محدود للنظام. في النظام المتسلسل تكون وضعية الذاكرة محددة من خلال حساب المعالج الوحيد، إذ أن جزء من الذاكرة لا يتغير لو لم يغيره المعالج. بينما في النظام الموزع، المعالج P_i لا يراقب إلا وضعه المحلي. المعطيات الموزعة على المعالجات الأخرى تتغير من دون أي مراقبة من P_i.

[8] : Cyril Gavoile , Algorithmes distribués , Université de bordeaux , (2015), P8.

- المزامنة (synchronism):
- النظام المتزامن(synchronous system): نفترض أن زمن التواصل مسقف أو بالأحرى لكل معالج ساعة داخلية ذات الخاصية التالية: "ارسال رسالة من v في الوقت t لجاره u يجب أن تصل في الوقت t+1 لـu. تكون دورة الحساب لكل معالج كما يلي:
  - ✓ 1: ارسال رسائل إلى معالج واحد أو العديد من المعالجات المجاورة.
  - ✓ 2: استقبال رسائل من معالج واحد أو من العديد من المعالحات المجورة.
  - ✓ 3: القيام بالحساب المحلي(الداخلي). يفترض أن الحساب الداخلي يأخد وقت قليل جدا أمام المرحلة 2. إذن دورة الحساب تأخذ كل الوقت في انتظار إستقبال الرسائل من المعالجات مما يؤدي إلى الإنسداد أحيانا.
- النظام غير المتزامن (asynchronous system) : عكس سابقه، الخوارزمي هنا موجه من خلال الأحداث ويكون:
  - عند الحدث E1 ، أفعل X1 (مع E1= وصول الرسالة.
  - عند الحدث E2 ، أفعل X2
- عدم التزامن و عدم التحديد(asynchronous and non– deterministic)

في نظام التتابع غير المحدد فهو مرتبط بالإستعمال العشوائي لخانات الاستعلام bits تستعمل فيه الدالة RANDOM للقيام بهذا الغرض، أما في نظام التتابع المحدد فبنفس المدخلات، نحصل دائما على نفس المخرجات. في النظام الموزع، لسنا بحاجة إلى الدالة العشوائية للحصول على عدم التحديد ، يمكن تنفيذ مرتين نفس الخوارزمي على نفس المدخلات ونتحصل على نتيجتين مختلفتين[9].

الشكل(1–5): التواصل بين المعالجات في النظام الموزع



المصدر : Cyril Gavoile ، مرجع سبق ذكره ، ص.3 ، طبعة 2005

[9]: Cyril Gavoile ، مصدر سبق ذكره ، ص6

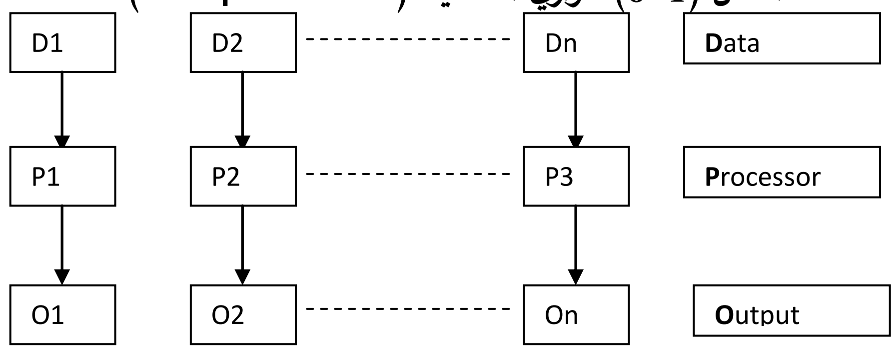
من خلال الشكل (1-5) ، نلاحظ أن المعالج Pv1 أرسل القيمة1 للمعالج Pu في حين بعث Pv2 القيمة صفر، فالقيمتان 1 و صفر كلاهما مدخلات للنظام ، و في مخرج النظام لا نعرف قيمة u هل تساوي 0 أم 1. وتكون النتيجة print(x) مختلفة عند كل تنفيذ. بمعنى ليس دائما لنفس المدخلات نفس المخرجات.

د- تعددية المهام(multitasking) : يمكن تعريف تعددية المهام على أنه القيام بعدة عمليات أو عدة مهام من طرف نظام التشغيل. حيث يمكنه تنفيذ أكثر من مهمة أو برنامج في نفس الوقت. تسمح تعددية المهام مثلا تنفيذ عدة برامج مختلفة و الطباعة على الطابعة و التخزين على القرص في الوقت نفسه دون انتظار انتهاء كل مهمة من هذه المهام قبل تنفيذ الأخرى كما هو معمول به في النظام المتسلسل. في الأجهزة المتعددة المعالجات تبنى الخوارزميات فيها على أساس تعددية المهام و التي تنفذ في وقت واحد حيث تقوم كل وحدة معالجة بمعالجة مسألة جزئية و لكن بشكل متزامن مع الوحدات الحسابية الأخرى و تكون النتيجة النهائية بجمع تلك النتائج المتعددة. و ترتكز البرمجة متعددة المهام على فهم طريقة تنفيذ التعليمات على البيانات [10].

2.1: مصادر و حدود التوازي : يمكننا الحصول على التوازي من مصادر مختلفة[11]:

1- توازي المعطيات(data parallelism): في توازي المعطيات نجد نفس العملية تنفذ من طرف كل معالج على معطيات مختلفة. و يمكن توضيح ذلك في الشكل(1-6)

الشكل (1-6): توازي المعطيات(data parallelism)



المصدر: Aurélia Marchand, 'Les architectures parallèles-MPI', p.7

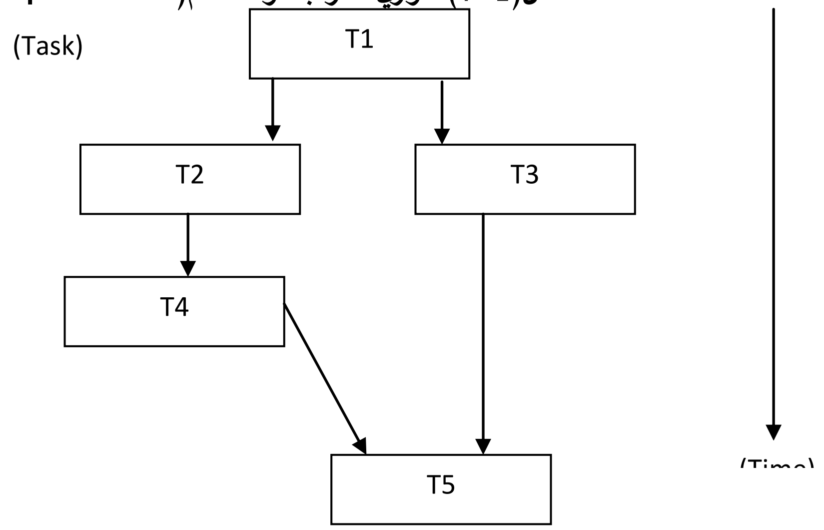
[10] : كندة زين العابدين، " **خوارزميات المعالجة المتوازية و برمجتها**"، جامعة دمشق، 2006 ، ص. 14

[11] : Aurélia Marchand, '**Les architectures parallèles**-MPI', p.7.

2- توازي المراقبة أو التحكم(control parallelism)

يتم هنا تحقيق العمليات على العديد من المعالجات و في نفس الوقت. يحتوي البرنامج على مقاطع أو أجزاء مستقلة و التي يمكن تنفيذها بالتوازي.

الشكل(1-7): توازي المراقبة أو التحكم(control parallelism)



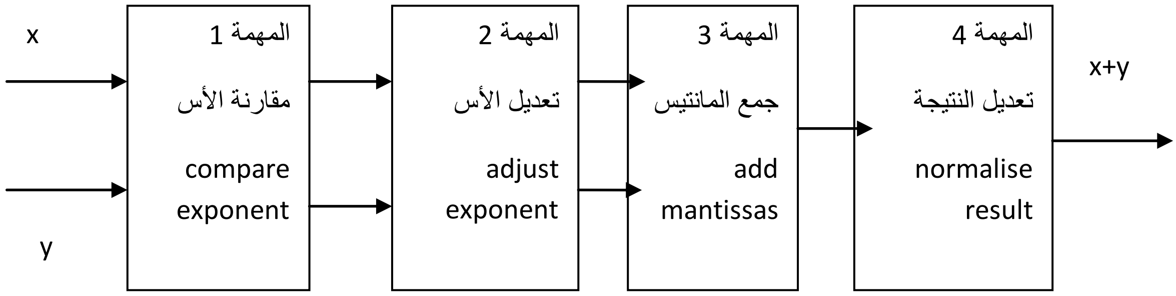
المصدر: Aurélia Marchand, '**Les architectures parallèles**-MPI', p.7.

3- توازي التدفق(flow parallelism): يمكن هنا اجراء العمليات الحسابية أو مهام بالتتابع على نفس التدفق للمعطيات و هو ما يسمى بالعمل بخط الأنابيب أو كذلك على مستويات(pipeline)[12]. تستعمل طريقة خط الأنابيب لزيادة كفاءة الحاسبات وحيدة المعالج و هذا بزيادة كمية حساب الوحدة الحسابية و المنطقية ، و كذلك في وحدة التحكم زيادة عدد التعليمات الواجب معالجتها في الدورة الزمنية للحاسب [13]. و لتوضيح العمل بخط الأنابيب نأخذ مثال أول و هو تنفيذ عملية جمع عددين حقيقيين x و y مكونة من أربع مهام أو مستويات الشكل(1-7-أ) و مثال ثاني لتنفيذ عملية حسابية مقسمة لخمس عمليات جزئية أو مهام و على معطيات مختلفة (D1, D2, D3, D4, D5, D6) كم هو مبين في الجدول(1-1).

[12]: Aurélia Marchand ، مصدر سبق ذكره ، ص8

[13] :Fayez Gebali,"Algorithms and Parallel Computing", University of Victoria, USA, 2011 , p.139

الشكل(1-7-أ): جمع عددين x+y من أربع مهام



المصدر: DANIEL C. Hyde ، مصدر سبق ذكره ، ص14

جدول(1-1): توازي التدفق(pipeline)

Time	Task1	Task2	Task3	Task4	Task5
t1	Data 1				
t2	D2	D1			
t3	D3	D2	D1		
t4	D4	D3	D2	D1	
t5	D5	D4	D3	D2	D1
t6	D6	D5	D4	D3	D2

أين: Time : هو زمن تنفيذ المهمة ، TASK: هي المهمة ، Data: هو معلم المهمة

المصدر: Aurélia Marchand ، مصدر سبق ذكره ، ص8

تعتمد طريقة خط الأنابيب على تقسيم عملية إلى عمليات جزئية أي مهام فرعية. تبدأ العملية في الوقت t1 على المعلم D1 من طرف المهمة رقم1، تليها مباشرة في الوقت t2 تنفيذ المهمة رقم 1 ولكن على المعلم D2. في هذه الأثناء يكون المعلم D1 تتم معالجته من طرف المهمة 2 و هكذا...تتم معالجة المعلم D1 عندما يصل إلى المهمة رقم 5 و يكون قد أستغرق t5 من زمن التنفيذ تليها المعلم D2 بعد t6 من زمن التنفيذ و هكذا لبقية المعلمات.

هناك قيود قد تواجهنا أثناء تصميم الخوارزميات المتوازية يجب أخذها بعين الأعتبار و هو ما يعرف بحدود التوازي على غرار[14]:

[14] : Aurélia Marchand, ‘**Les architectures parallèles**-MPI’, p.9.

1− أرتباط المعطيات(data dependency):نوضح ارتباط المعطيات في المثال التالي[15]:

ارتباط المخرج(output dependency)	ارتباط التدفق(flow dependency)	
a = b + c	a= b + c	a= b + c
a= d + e	b = d + e	d= a + e
هناك ارتباط المعطيات في المخرج a و بالتالي فالتعليمتين ليست مستقلتين. لا يمكن تنفيذهما في نفس الوقت.	هناك ارتباط في المعطيات b في التعليمتين و بالتالي فالتعليمتين ليست مستقلتين. لا يمكن تنفيذهما في نفس الوقت.	هناك ارتباط بين التعليمة الأولى و الثانية a. لا يمكن تنفيذهما في نفس الوقت.

2− ارتباط التحكم:لدينا المثال التالي:

التعليمات I1 و I3 مستقلتان و لكن حسب قيمة a ، I3 يمكن تنفيذها أولا و بالتالي هناك ارتباط بين I3 و I4.	I1 : a = b + c ; I2 : if (a < 0) I3 : { d= e + f ; } I4 : g= d + h ;

3− ارتباط في الموارد(dependence of resources) :يكون الارتباط في الموارد عندما يكون عدد المعالجات غير كاف لتنفيذ التعليمات المتوازية و هي مستقلة بعضها عن بعض.

[15]: Aurélia Marchand. ، مصدر سبق ذكره ، ص9

4- نسبة زمن التواصل مع زمن الحساب: يكون في بعض الأحيان التوازي عملية غير مفيدة إذا كان زمن التواصل بين المعالجات كبيرا جدا مقارنة بزمن الحساب و يؤدي بالتالي إلى زيادة في زمن التنفيذ.

3.1 الغرض من استخدام التوازي:

إن الدوافع التي أدت إلى استعمال التوازي في الحاسوب هو عدم إمكانية زيادة سرعة المعالجة كما هو معروف في قانون مور Loi de Moore و التي تنص على أن سرعة المعالج تتضاعف كل 18 أشهر لأنها فزيائيا غير ممكنة لوصولها لحدها الأقصى . لذا تم التوجه لتقنيات أخرى لزيادة أداء الحاسب على غرار تعدد النوى في المعالج الواحد أو تصميم حاسوب متعدد المعالجات.و بالتالي كان الحساب المتوازي هو الحل الوحيد لمعالجة الحسابات الكبيرة في وقت قصير. كما يمكن للحساب المتوازي إيجاد الحل للعديد من المسائل المعقدة في وقت معين[16] .

4.1 دراسة المعالجة المتوازية:

أحدثت المعالجة المتوازية مؤخرا ثورة علمية كبيرة في مجال الحواسيب و بدأت تدخل العالم كل يوم عن طريق معالجة المعطيات في شكل قواعد معطيات موزعة. و أصبح من الضروري للمبرمجون فهم مبادئ المعالجة المتوازية لكي يستطيعوا برمجة حواسيب المستقبل[17].

و جميع الحواسيب العملاقة (Supercomputers) هذه الأيام تعتمد بشكل كبير على التوازي. و هي تُستخدم في مستوى البرمجيات و كذلك التصميم الهندسي للعتاد (Hardware). و قد اشتد السباق و التنافس بين دول العالم اقتصاديا مما تطلب منها القيام باكتشافات علمية في علم الحاسوب ليقوموا بتوظيف الحواسيب العملاقة بشكل فعال.

و تعتبر المعالجة المتوازية حقلا جزئيا من علم الحاسب و التي تتضمن مفاهيم و أفكاراً من علوم الحاسوب النظرية و بنية الحاسب و لغات البرمجة و الخوارزميات و مجالات التطبيق مثل الذكاء الاصطناعي و الرسوم[18].

[16] : Laurence Very& Violene Louvet, "Géneralitées sur le parallelisme", p.15

[17]: DANIEL C. Hyde، مصدر سبق ذكره ، ص8

[18]: DANIEL C. Hyde ، مصدر سبق ذكره ص9

5.1 مجالات تطبيق المعالجة المتوازية:

يتم اللجوء للمعالجة المتوازية في التطبيقات التي تتطلب قدرة حسابية و تخزينية عالية للحاسب. و من بين هذه التطبيقات نذكر مثلا: علوم الكون و المحيط و التنبؤات المناخية. و في مجال الهندسة دراسة ديناميكية السوائل، معالجة الصور ، الذكاء الاصطناعي، النمذجة و المحاكاة إذ يصبح الحاسوب عبارة عن مخبر افتراضي غير مكلف لإجراء التجارب ذات الأحجام الكبيرة من المعطيات و التي لا يمكن إجرائها في الحقيقة و استخلاص نتائجها في أقصر وقت. من هذه التطبيقات ما تعد خطيرة كالصيانة النووية أو صيانة الطائرات أو الكشف الطبي. و من تطبيقات المعالجة المتوازية كذلك علوم قواعد المعطيات، التشفير ، العلوم المالية و الإقتصادية خاصة منها مسائل الأمثلية إلى غير ذلك [19].

6.1 الحاسبات المتوازية:

الحاسبات المتوازية هي آلات لها عمارة متوازية تحتوي على العديد من المعالجات قد تكون متشابهة أو مختلفة تعمل بشكل متزامن لمعالجة مسألة معينة. تتميز بنيتها بعدم **محدودية الذاكرة و عدد** المعالجات، تسريع الحسابات المعقدة و المكلفة بالنسبة لوقت المعالج (حساب المصفوفات ذات الأحجام الكبيرة، المحاكاة العددية،.....) و باستقلالية المعالجة[20].

يمكن تصنيف ذاكرات الحاسبات المتوازية إل قسمين:

1. 6 .1 الحاسبات المتوازية ذات ذاكرة مشتركة:

تتميز هذه الحاسبات بساعة داخلية مستقلة لكل معالج، و لكن ذاكرة مشتركة بين هذه المعالجات أين يمكن لها القراءة و الكتابة في نفس الموقع من الذاكرة.مما يمكن من تحقيق التوازي التعليمات و المعطيات. المبرمج هنا يمكنه فقط تحديد جزء البرنامج الذي يتم تنفيذه من هذا المعالج أو ذلك بالإضافة لإدارة التزامن بينها[21]. لكل معالج ذاكرة محلية خاصة به، و يمكنه العمل بكل استقلالية عن الأخرين و لكن كل تغيير في محتوى موقع الذاكرة الشاملة من طرف معالج ، يكون مرئيا من طرف الآخرين[22].

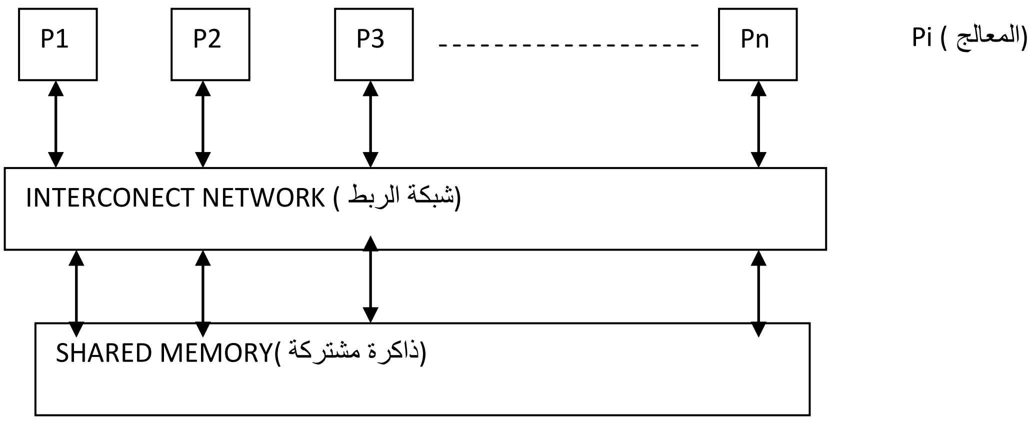
[19] : Eric Goubault & Sylvie Putot," Calcul Parallèle et Distribué", 2014, p.5

[20] : Aurélia Marchand، مصدر سبق ذكره ، ص6

[21] :Rédha LOUCIF, "Parallélisation d' Algorithme Combinatoire", 2014, p32

[22] :Violaine Louvet, "Architectures des ordinateurs, Concepts du Parallélisme", ICJ-CRS, 2010, p87

الشكل رقم(1-8): الحاسبات المتوازية ذات الذاكرة المشتركة

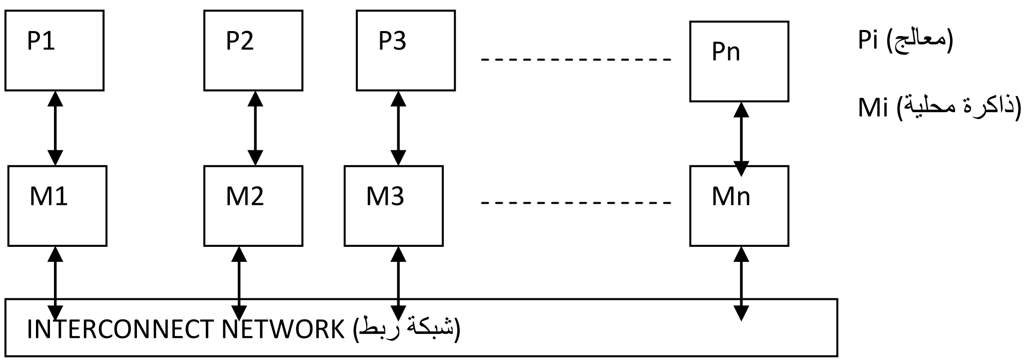


المصدر : Aurélia Marchand، مصدر سبق ذكره ، ص17

1. 6. 2 الحاسبات المتوازية ذات ذاكرة موزعة:

في هذا النوع من الحاسبات كل معالج له ذاكرته المحلية ، ينفذ تعليمات مشابهة أو لا للمعالجات الأخرى. تكون العقد المختلفة مرتبطة فيما بينها بواسطة شبكة الربط، و يتم التواصل بين مختلف المعالجات من خلال تبادل الرسائل. و تتميز هذه الحاسبات بسهولة زيادة عدد المعالجات بها و لكن بالمقابل صعبة البرمجة[23]. في هذا النوع من الحاسبات ليس هناك مفهوم للذاكرة الشاملة بين المعالجات، و كل تغيير في ذاكرة محلية لمعالج ليس لها تأثيرا في ذاكرة المعالجات الأخرى، وإذا كان معالج في حاجة لمعطيات من ذاكرة معالج آخر، فإن المبرمج هو الذي يعرف و يحدد التواصل بينهما.

الشكل رقم(1-9): الحاسبات المتوازية ذات الذاكرة الموزعة



المصدر : Aurélia Marchand، مصدر سبق ذكره ، ص17

[23]: Rédha LOUCIF ، مصدر سبق ذكره ، ص33

1. 6. 3 الحاسبات المتوازية ذات الذاكرة الهجينة:

إن أقوى الحاسبات في العالم الحالي تحتوي على خليط من الذاكرات المشتركة و الموزعة، و تكون العقد حجر الأساس لها، و كل عقدة منها عبارة عن حاسب متعدد المعالجات ذي ذاكرة مشتركة، ترتبط هذه العقد فيما بينها عبر شبكة الربط[24].

يدخل ضمن فيئة الحاسبات ذاكرة مشتركة، الحاسبات متعددة المعالجات التناظرية (SMP) والحاسبات ذات المعالجات متعددة النوى. و في فيئة الحاسبات ذاكرة موزعة نجد الحاسبات متعددة المعالجات المتوازية (MPP) و الحاسبات (Clusters)[25].

1. 7 نماذج الحاسبات المتوازية:

تم اقتراح عدة نماذج للحاسبات المتوازية من طرف الباحثين لحل مختلف المسائل. فالنموذج RAM (Random Access Machine) Von Neumann و المستعمل على نطاق واسع في دراسة الخوارزميات المتسلسلية و يتمثل في وحدة معالجة مركزية و ذاكرة ذات ولوج عشوائي (RAM).إن نجاح هذا النموذج يعود أساسا إلى بساطته و سهولته في ابراز نجاعة الخوارزميات المتسلسلة على الآلات المتسلسلة. في الحساب المتوازي، النموذج PRAM (Parallel Random Machine) هو الأكثر إستعمالا و هذا بالرغم من عدم وجود آلة حقيقية له، ولكن يسمح لمصمم الخوارزميات بالتركيز على الخصائص الأساسية للمسألة و عدم الغوص في التفاصيل. في شكله البسيط، النموذج يفترض مجموعة من المعالجات و ذاكرة شاملة مشتركة لتنفيذ نفس البرنامج بصفة متزامنة، أي يتم تنفيذ نفس السلسلة من التعليمات و لكن على معطيات مختلفة. يفترض في النموذج أن الولوج للذاكرة يكون ثابت و هذا مهما يكون عدد المعالجات بالرغم من أن هذا الإفتراض لا يعبر عن الحقيقة و لكن يبقى هذا النموذج الأكثر ملائمة لدراسة الخوارزميات المتوازية و تقييم نجاعتها من الناحية النظرية[26].

1. 8 قياس نجاعة الخوارزميات المتوازية:

يتم تقييم نجاعة الخوارزمي المتسلسل بقياس زمن تنفيذه [27].

---

[24]: Violaine Louvet ، مصدر سبق ذكره ، ص88

[25] :Pierre Fraigniaud, "Algorithmique parallele et distribuée", Univ. Paris Diderot, 2017, p47

[26] :Pierre Delisle,"parallélisation d'un algorithme d'optimisation par colonies de Fourmis", 2002, p17-18

[27] Jun Zhang, **"Parallel computing: Performance and scalability"**, University of Kentuckey, p.2

في الخوارزمي المتوازي يكون التقييم أصعب و معقد لأن النجاعة مرتبطة بحجم المدخلات n و كذلك بعدد المعالجات p و الآلة التي يتم تنفيذ البرنامج عليها. لذا نلجأ إلى حساب معايير أخرى كحساب التسريع و الفعالية.

أ–التسريع (Speedup) (S_P): تسريع برنامج متوازي هو حاصل قسمة زمن التنفيذ الخوارزمي التسلسلي (t_s) على زمن تنفيذ الخوارزمي المتوازي (t_p) لحل نفس المسألة و على حاسوب متوازي معين [28]. كما يسمح قياس التسريع بحساب نسبة الحساب المتوازي على أفضل خوارزمي متسلسل.

و من ذلك نكتب: $\mathbf{s_p= t_s/t_p}$

يكون التسريع في أعلى قيمة (max) عندما تكون تساوي عدد المعالجات p. أي عندما يكون العمل المحقق من طرف الخوارزمي التسلسلي قد تم توزيعه على مختلف المعالجات و لم ينتج عنه أي تكلفة إضافية. و لهذا يمكننا أن نلاحظ في عبارة التسريع ارتباطها بعدد المعالجات و بالتالي يكون من الضروري التطرق لمفهوم الفعالية أو نسبة النشاط [29].

ب– الفعالية (efficiency) (E_p): $\mathbf{E_p =S_p/p= t_s/(p*t_p)}$ تترجم نسبة استعمال ل p معالج، و تسمح بقياس مردود الحساب المتوازي. نظريا الفعالية هو عدد موجب $0\leq E_p \leq 1$.

نقول: – إذا ارتفع زمن التنفيذ المتوازي t_p ، انخفض التسريع s_p .

– إذا انخفض زمن التنفيذ المتوازي t_p ، ارتفع التسريع s_p .

– هنا كذلك نلاحظ أنه كلما ارتفع عدد المعالجات انخفظت الفعالية و العكس صحيح.

$$E(p) = t_s/c_t(p) = t_s/pt_p(p) = S(p)/p \leq 1$$

حيث $c_t(p)$ هي الكلفة الزمنية الاجمالية لجميع المعالجات.

$$c_t(p) = p\ t_p(p)$$

– و عمليا الفعالية قد تكون أكبر أو تساوي واحد و هذا في الحاسبات فائقة السرعة(superscalar computer).

---

[28] : Jun Zhang, **"Parallel computing: Performance and scalability"**, University of Kentuckey, p.4

[29] Rédha Loucif: مصدر سبق ذكره ص41

$$S(p) \leq 1/\sigma \Rightarrow E(p) = S(p)/p \leq 1/(\sigma p)$$

- حيث σ هو الجزء من العبء الغير قابل للتوازي في الخوارزمي.

ج- حساب الكلفة الزمنية الإجمالية(TOTAL COST): يمكن حساب الكلفة الإجمالية كما يلي[30]:

-الكلفة الزائدة(parallel overhead) :و تحسب كما يلي:

$$t_o(p) = pt_p(p) - t_s$$

حيث:$pt_p(p)$ هو زمن تنفيذ جميع المعالجات p (أو الكلفة الاجمالية) .

t_s هو زمن تنفيذ الخوارزمي المتسلسل.

يكون الحساب المتوازي عادة أكبر كلفة من الحساب المتسلسل. و يمكن البرهان عليه كما يلي:

$$t_p(p)= t_{fix} + t_{com} + t_{par} /p \Rightarrow pt_p(p)= pt_{fix}+pt_{com}+t_{par}+t_{fix}-t_{fix}$$

$$= (p-1)t_{fix} + pt_{com}+t_{fix}+ t_{par} > 0 + t_s \Rightarrow t_o(p) >0$$

1 .9 تكلفة العمل في الخوارزمي المتوازي:

تنقسم تكلفة العمل في الخوارزمي المتسلسل (w_{seq})إلى قسمين. قسم ثابت (w_{fix}) و قسم قابل للتوازي[31](w_{par}).

$$w_{seq} = w_{fix} + w_{par}$$

وتكون تكلفة العمل إذن على خوارزمي متوازي و على p معالج:

$$w_p(p)= w_{fix} + w_{par} /p$$

- في برنامج متوازي حقيقي، يجب أخذ في الاعتبار تكلفة العمل w_{com} الناتجة عن التزامن بين المهام (Tasks synchronisation).

$$w_p(p) = w_{fix} + w_{par} /p + w_{com}$$

- **زمن التنفيذ(elapsed time)**

هو زمن الحساب المتوازي $t_p(p)$ من البداية حتى نهاية آخر مهمة في البرنامج من طرف p معالج .

[30] : article « Evaluation et critéres de performance d’un calcul paralléle https://repo.zenk-security.com/Others/Evaluation%20et%20criteres%20de%20performances%20d.un%2, p.2-

[31]: article « Evaluation et critéres de performance d’un calcul paralléle، مصدر سبق ذكره ، ص4

$S(p) \leq p$: بعبارة أخرى، الحساب المتوازي على p معالج لا يمكن تنفيذه p مرة أسرع من أفضل خوارزمي متسلسل.

البرهان[32] :

$$t_s/t_p \leq p \Rightarrow t_s \leq p\ t_p = (t_{fix} + t_{par})/(t_{fix} + t_{par}/p + t_{com})$$

$$=p(t_{fix} + t_{par})/(pt_{fix} + t_{par} + pt_{com} + t_{fix} - t_{fix})$$

$$= p(t_{fix} + t_{par})/[(t_{fix} + t_{par})+(p-1)t_{fix} + pt_{com}]$$

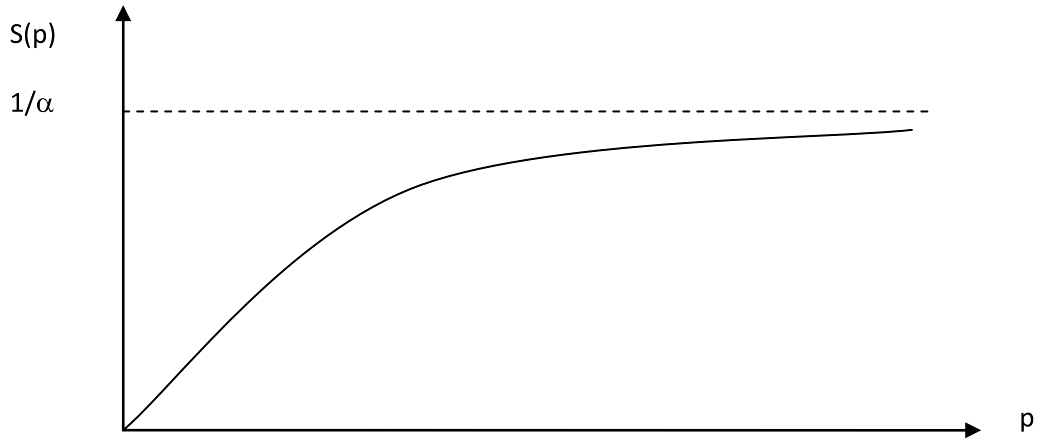
$$= p/[1+ (p-1)t_{fix}/(t_{fix}+t_{par}) + p_{com}/(t_{fix}+t_{par})]$$

$$= p/(1+ k) \leq p; \ k \geq 0$$

و من جهة أخرى فحسب قانون أمدال(loi d'Amdahl) لدينا $s(p) \leq 1/\alpha$ حيث $\alpha= t_{fix}/t_{seq}$ يمثل الجزء الغير قابل للتوازي أو بعبارة أخرى فتسريع الخوارزمي يكون محدود بثابت α وهذا مهما كان عدد المعالجات p.

$$S(p) = 1/[1/p +(1-1/p)\alpha] \rightarrow 1/\alpha \ ; \ p \rightarrow \infty$$

الشكل(1−10):منحنى التسريع Sp بدلالة عدد المعالجات p.



المصدر: Evaluation et critéres de performance d'un calcul paralléle مصدر سبق ذكره ص 12

و من جهة أخرى كلما زاد عدد المعالجات كلما زاد زمن التواصل بيها($t_{com} \rightarrow \infty$ عندما $p \rightarrow \infty$).

$$S(p)= 1/[\alpha + t_{com}(p)/t_{seq}] \ ; \ s(p) \rightarrow 0 \ ; \ p \rightarrow \infty$$

[32] : article « Evaluation et critéres de performance d'un calcul paralléle

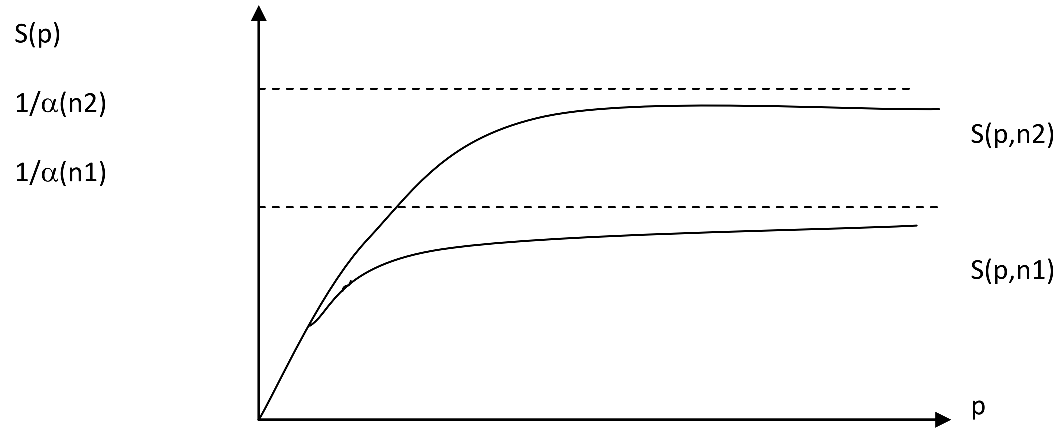
- **ارتباط التسريع بحجم المعطيات**

لا يرتبط التسريع بعدد المعالجات فقط بل كذلك بحجم المعطيات المعالجة (n):

$$W_{seq}(n) = w_{fix} + w_{par}(n)$$

بصفة عامة $w_{par}\uparrow$ عندما $n\uparrow$ و بالتالي $\alpha(n) = (w_{fix} / w_{seq}) \rightarrow 0$ و بالتالي عند عدد مثبت من المعالجات، التسريع يكون أحسن عندما يكون حجم المعطيات المعالجة أكبر.

الشكل(11-1):منحنى التسريع Sp بدلالة عدد المعالجاتp وحجم المعطيات n .



المصدر: **Evaluation et critéres de performance d'un calcul paralléle مصدر سبق ذكره ص 12**

- **توازن الأعباء (balance of workloads)** : يعرف على أنه قياس نوعية توزيع عبىء العمل على المعالجات[33].

$$\epsilon(p) = (w_{max} - w_{moy})/w_{max}$$

حيث:

W_i: هو عبىء العمل على المعالج p_i

W_{max}: عبىء العمل الأقصى $w_{max}= \max\{w_i\}$

W_{tot}: العبىء الكلي $w_{tot} = \sum w_i$

W_{moy}: معدل العبىء $w_{moy}=w_{tot}/p$

يكون التوازن تام عندما يكون $w_{max} = w_{moy}$

[33]: **Evaluation et critéres de performance d'un calcul paralléle ، مصدر سبق ذكره ، ص17**

شروط النجاعة : لحل نفس المسألة، تكون كلفة الحساب المتوازي أكبر من تكلفة الحساب المتسلسل[34].

السؤال: ما هو الشيء الذي يؤدي إلى هذه الزيادة في الكلفة(additional cost)؟

وكيف يمكن تقليص هذه التكلفة؟فإذا أخذنا العمل w_{fix} هو نفسه في الحساب المتسلسل أو الحساب المتوازي. بينما العمل w_{com} ينتج عن المعالجة المتوازية و من خصائصه أنه يزداد كلما أرتفع عدد المهام المتوازية و كذلك كلما ارتفع حجم معطيات المعالجة.

أ- التواصل:

في الغالب للتواصل بين المهام تأثير كبير في هذه الزيادة في العمل المتوازي .لأن عند التواصل لا يتم الحساب و بصفة عامة كذلك التواصل يتطلب وقت أكبر من الحساب. الحل يكمن إذن في التقليل من عدد التواصلات الممكنة و كذلك تغطية هذه التواصلات بالحساب(أي إجراء العمليات الحسابية عند التواصل).

ب- عدم توازن الأعباء(imbalance workload):

لعدم توازن الأعباء تأثير في زمن الحساب المتوازي. في التوازن التام ليس هناك مضيعة للوقت. بمعنى عدم توازن الأعباء يؤدي إلى الانتظارو بدون فائدة للمعالجات أقل عبأ.

ويمكن توضيح مفهوم التسريع من خلال المثال التالي:

نفترض أننا نريد بناء جدار، و أن كل عامل يحتاج إلى وحدة زمن واحدة لبناء الجدار ، فما هو الزمن اللازم ل n عامل للقيام بهذه المهمة. لنفرض كذلك أن الحالة المثالية هي أن عامل بناء لا يعيق زميله في العمل ، ففي هذه الحالة يجب على البنَّائين أن ينهوا بناء الجدار خلال $1/n$ وحدة زمن[35].

و بالتالي فإن نسبة التسريع في هذه الحالة تكون : $s_p=1/(1/n)=n$

و بالتالي ، فإن n عامل من عمال البناء أسرع n مرة من العامل الواحد.

10.1 أنواع المعالجة المتوازية:

المعالجة المتوازية هي إحدى أشكال معالجة المعطيات ، تسمح بتنفيذ عدد من الأحداث المتزامنة في نفس الوقت. هذه الأحداث المتوازية يمكن أن تكون على مستويات مختلفة[36]:

[34]: **Evaluation et critéres de performance d'un calcul paralléle ، مصدر سبق ذكره ، ص19**

[35]: DANIEL C. Hyde: : ، مصدر سبق ذكره ، ص11

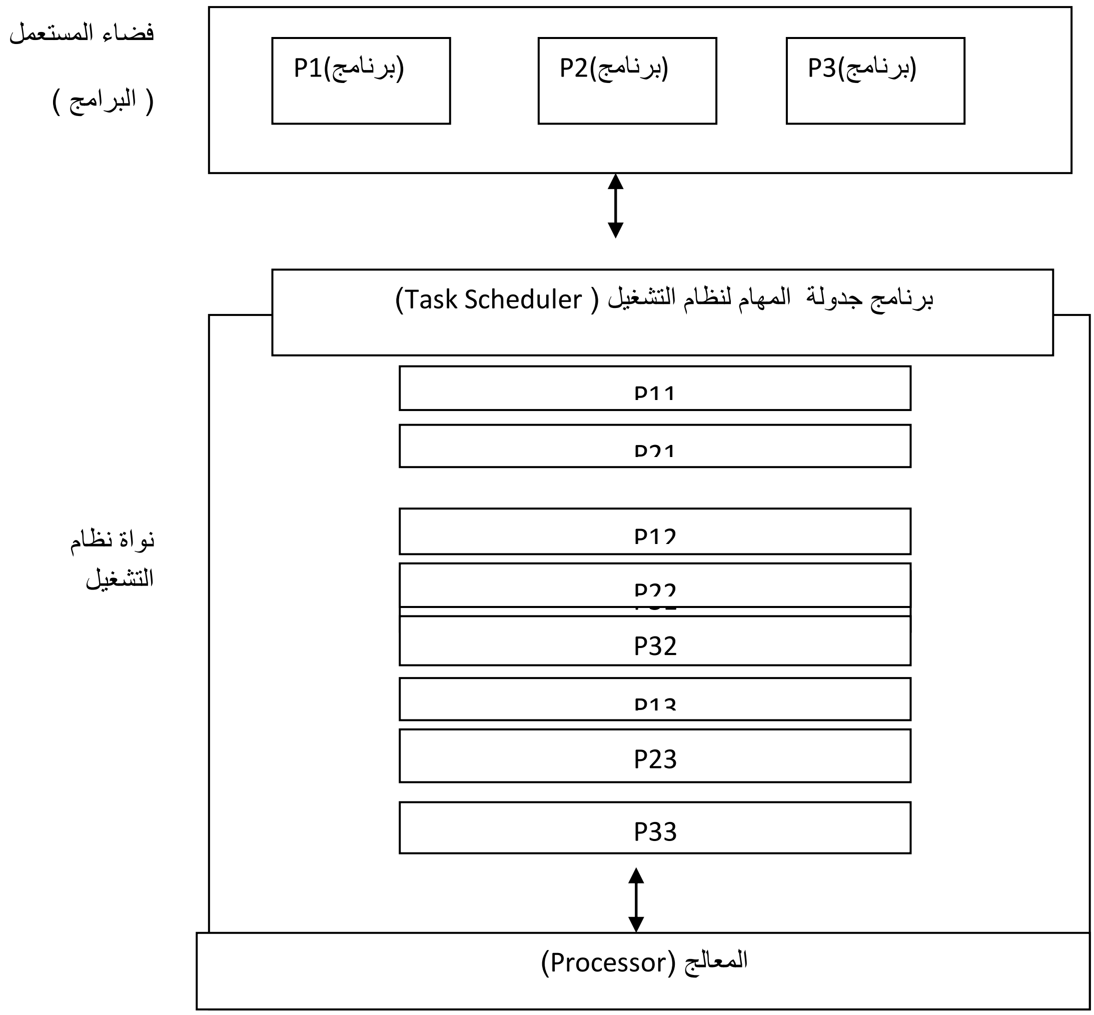
[36]: M. Eleuldj, "Architectures paralleles", EMI, 2014, p12

1.10.1 مستوى البرامج (Programs) :

يمكن تنفيذ هذه البرامج المستقلة عن بعضها في نفس الوقت باستعمال حاسوب متعدد المعالجات، كما يمكن تنفيذها على حاسوب من معالج واحد و هذا باستعمال قواعد تعدد البرمجيات و نظام المشاركة في وقت المعالج. وقد تكون هذه البرامج لمستعمل واحد أو للعديد من المستعملين للحاسوب. يقوم نظام التشغيل بتنفيذ جزء من البرنامج الأول ثم جزء آخر من البرنامج الثاني و بعد ذلك الرجوع لجزء ثاني من البرنامج الأول فجزء ثاني من البرنامج الثاني هكذا ..مما يوهم المستعمل أن جميع هذه البرامج تنفذ في نفس الوقت. في الحقيقة تعتمد الطريقة على تقسيم وقت الوحدة المركزية إلى مجالات زمنية متساوية صغيرة جدا حيث تشغل الوحدة المركزية البرامج المختلفة دوريا خلال هذه المجالات الزمنية المختلفة. في التنفيذ التسلسلي يتم تنفيذ البرنامج الأول فالثاني ثم بعد ذلك الثالث. أما في التنفيذ المتزامن فإن نظام التشغيل (برنامج الجدولة) يوضح أجزاء هذه البرامج في طابور قصد تنفيذها من طرف المعالج. هذه الأجزاء تكون في إحدى الوضعيات(في حالة توقف ، في حالة مقبل على التنفيذ أو في حالة التنفيذ). يتم تقديم هذه البرامج الموجودة في الطابور للتنفيذ حسب نوع برنامج الجدولة في نظام التشغيل(جدولة وقائية ، جدولة تعاونية). و فيما يلي الشكل(1−12) لتضيح أكثر[37].

[37] :Site https://fr.wiki.books.org,"systeme d'exploitation et multiprogrammation", version PDF, p.7

الشكل (1−12): تعددية البرمجيات ونظام المشاركة في الوقت



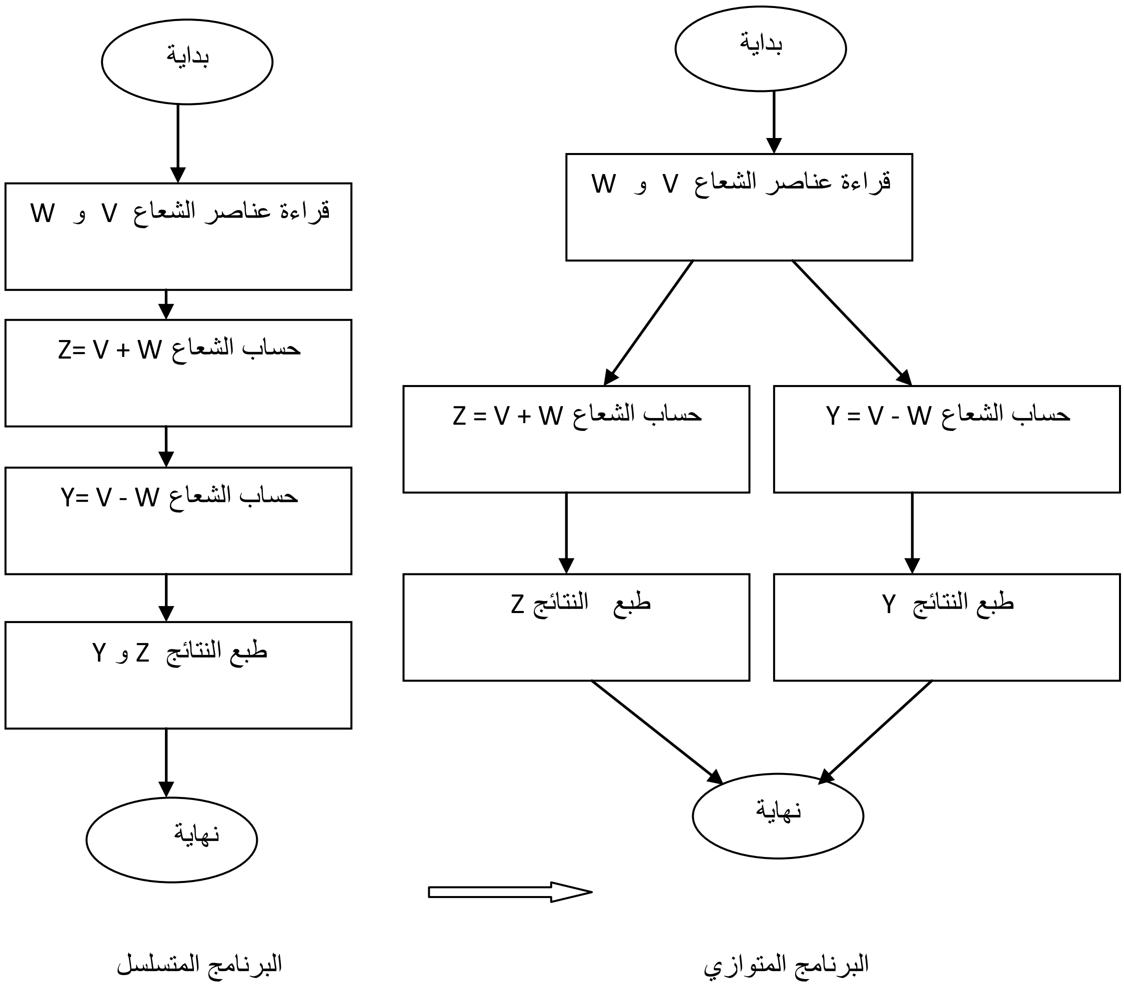
المصدر: Site https://fr.wiki.books.org ، مصدر سبق ذكره ، ص8

2.10.1 مستوى الإجرائية: (Procedure) : يتكون البرنامج عادة من إجرائيات ، و كل إجرائية تتكون بدورها من تعليمات. قد تكون بعض هذه الاجرائيات بينها علاقة ارتباط. بمعنى تكون مخرجات إجرائية معينة هي مدخلات لإجرائيات أخرى. أو قد تكون بعض هذه الإجرائيات مستقلة عن بعضها ، و بالتالي يمكن تنفيذها بالتوازي على معطيات و بواسطة معالجات مختلفة. يقوم المبرمج بتحديد هذه الإجرائيات المستقلة و هذا طبعا بعد تحليلها و التأكد من عدم ارتباط المعطيات بينها. كما لنظام التشغيل دور مهم في إدارة ها المستوى من المعالجة. يبين الشكل(1−13) التالي هذا المستوى من التوازي:

يقوم هذا البرامج بتنفيذ عمليتين على الأشعة ، الأولى: جمع عناصر كل من الشعاع V و W و النتيجة توضع في الشعاع Z و الثانية عبارة عن طرح لعناصر W من عناصر الشعاع وV و النتيجة توضع في الشعاع Y. و بالتالي يقوم البرنامج بالاجرائيات التالية كما هو مبين في الشكل (1−13):

- قراءة عناصر الشعاع V و عناصر الشعاع W .
- جمع عناصر الشعاع V و عناصر الشعاع W و وضع النتائج في الشعاع Z.
- طرح عناصر الشعاع W من عناصر الشعاع V و وضع النتائج في الشعاع Y.
- طباعة النتائج: الشعاع Z و الشعاع Y.

الشكل (1−13): تحويل برنامج تسلسلي إلى متوازي (مستوى الاجرائيات)

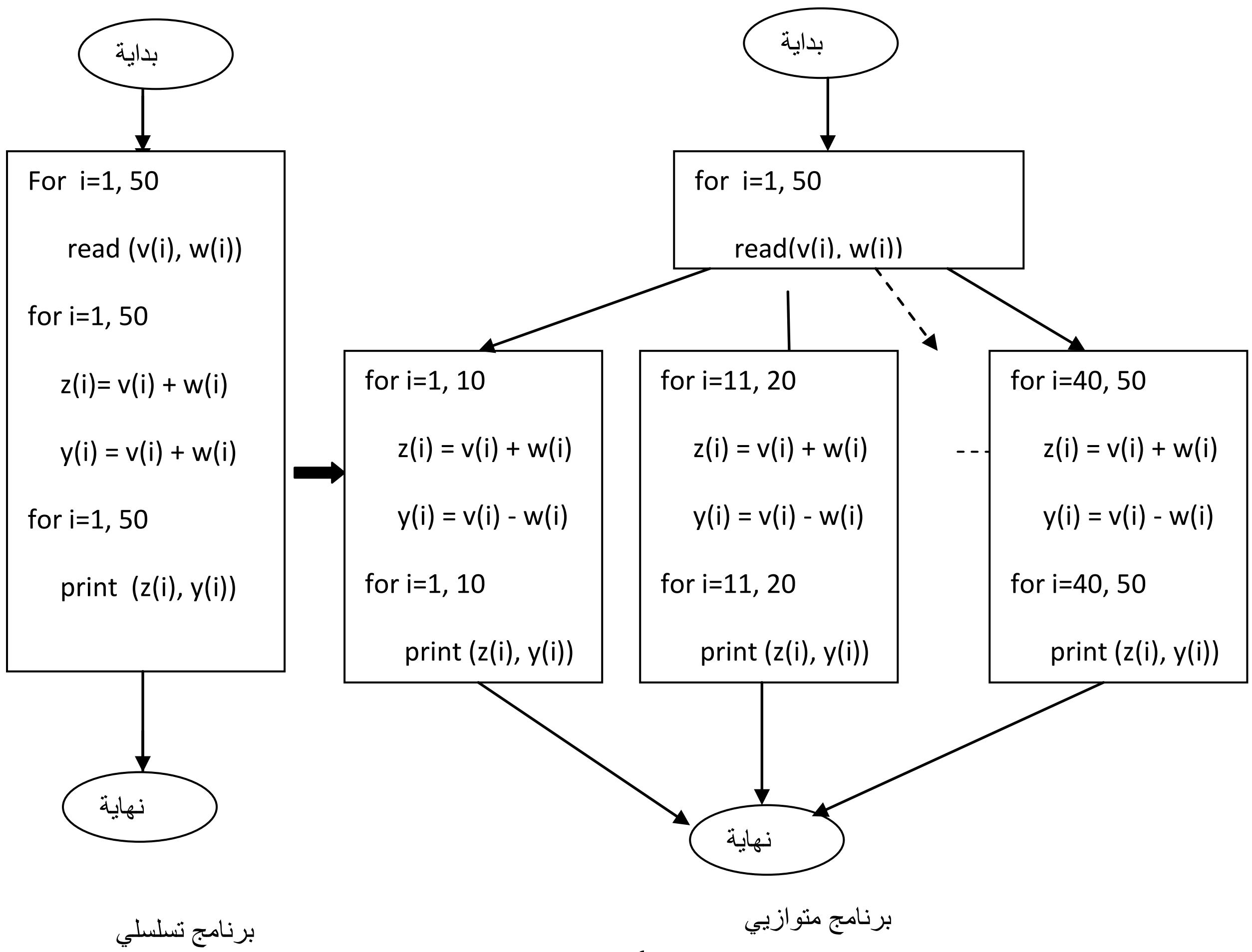


المصدر :من إعداد الطالب

نلاحظ في الشكل(1−13) أن العمليات Z= V + W و Y= V− W و ليس بينها تبعية، أي مستقلتان و بالتالي يمكن تنفيذهما بالتوازي. لذا يمكن تحويل البرنامج التسلسلي إلى برنامج يحتوي على أجزاء يمكن تنفيذها بالتوازي على معالجات مستقلة.

3.10.1 مستوى التعليمات (Instructions) : إن تحليل التعليمات يمكن المبرمج من التأكد من وجود أو عدم وجود الإرتباط فيما بينها. فالتعليمات غير المرتبطة فيما بينها يمكن تنفيذها على معطيات و من طرف معالجات مختلفة على التوازي. ففي الشكل(1−14) مثلا فالتعليمة Z(I)= V(I) + W(I) و التعليمة Y(I)= V(I) − W(I) يمكن حسابهما بدون أن تؤثر الواحدة في الأخرى. و يبين الشكل (1−14) تحويل برنامج تسلسلي إلى متوازي على مستوى التعليمات. إذ نلاحظ أن تكرارات التعليمة for مستقلة عن بعضها وأن حساب التعليمات في التكرار السابق لا يؤثر في حسابها في التكرار الموالي و بالتالي يمكن تقسيم تكراراتها إلى أجزاء و تنفيذها بالتوازي على معطيات و معالجات مختلفة. نجد هذا النوع من التوازي خاصة في معالجة مسائل الأشعة و المصفوفات.

الشكل رقم (1−14) : تحويل برنامج تسلسلي إلى متوازي (مستوى التعليمات)



4.10.1 مستوى التعليمة (Instruction): يعتمد في البحث عن إمكانية إستغلال التوازي على مستوى التعليمة أو العملية إلى تقسيم هذه الأخيرة إلى أجزاء عمليات متتالية (على مراحل)، بحيث يمكن تنفيذها على التوازي و على معطيات مختلفة. و تعرف هذه الطريقة بتسمية خط الأنابيب ، إذ تكون مخرجات مرحلة مدخل لمرحلة أخرى. و لتوضيح أكثر نأخذ المثال(1−1). ففي الجدول (1−2) مثلا نرى عملية ضرب لعددين A(2)*B(2) لا تنطلق في التنفيذ إلا بعد أنتهاء العملية A(1)*B(1) و هذا هو المعمول به في المهام المتسلسل. في حين نرى في الجدول(1−3) أن حساب عملية A(2)*B(2) تنطلق مباشرة مع المرحلة الثانية لعملية حساب A(1)*B(1) و بالتالي تنفيذ هذه المراحل بالتوازي عل معطيات مختلفة و ينتج عن ذلك تقليص في زمن تنفيذ العمليتين.

مثال(1−1): عملية ضرب عددين في الحاسبات الشعاعية:نوضح في هذا المثال كيفية ضرب عددين في الحاسبات الشعاعية[38].

```
Do i=1,n
  C(i)=A(i)*B(i)
Enddo
```

التعليمة C(i)=A(i)*B(i) تتطلب عدة عمليات جزئية أو مراحل لتنفيذها(E1: مقارنة القوى ،E2: وضع العددين بنفس القوة، E3: ضرب العددين، E4: تعديل النتيجة، E5: كتابة النتيجة).

في الحالة الأولى: التنفيذ على حاسوب تسلسلي للعمليتين : C(1)=A(1)*B(1); C(2)=A(2)*B(2)

[38]: Eric Goubault &Sylvie Putot: ، مصدر سبق ذكره ، ص4

الجدول (1-2) : تنفيذ عملية الضرب على حاسوب تسلسلي(scaler)

الأعداد	**العمليات الجزئية (مراحل حساب العملية)**									
A(1), B(1)	E1	E2	E3	E4	E5					
A(2), B(2)						E1	E2	E3	E4	E5
result					C(1)					C(2)

المصدر: Eric Goubault &Sylvie Putot,"calcul parallele et distribué", Ecole polytechnique, PARIS-SACLAY, 2014, p.4

في الحالة الأولى و كما هو مبين في الجدول(1-2) ، إذا كان تنفيذ كل مرحلة مثلا يتطلب دورة زمنية(clock cycle) ، فبالتالي يجب 5 دورات لتنفيذ تعليمة و 10 دورات لتنفيذ تعليمتين و هكذا...

في الحالة الثانية: التنفيذ على حاسوب شعاعي(vectoriel) C(1)=A(1)*B(1); C(2)=A(2)*B(2)

الجدول (1-3): تنفيذ عملية الضرب على حاسوب شعاعي(خط الأنابيب)

الأعداد	العمليات الجزئية (مراحل حساب العملية)					
A(1), B(1)	E1	E2	E3	E4	E5	
A(2), B(2)		E1	E2	E3	E4	E5
result					C(1)	C(2)

المصدر: Eric Goubault &Sylvie Putot ، مرجع سبق ذكره، ص.4

في الحالة الثانية، في العمل بخط الأنابيب كما هو مبين في الجدول(1-3) ، يتطلب تنفيذ تعليمة واحدة 5 دورات زمنية و 6 دورات زمنية لتنفيذ العمليتين . و بالتالي هناك زيادة في سرعة التنفيذ.

1 .11 التطور التاريخي للحاسبات:

يمكن تقسيم فترات تطور الحاسبات بحسب التطور الذي طرأ على الدوائر الكهربائية للحاسب و طريقة عمله ، و لغات البرمجة التي يتم فيها التواصل مع الحاسب. و قد رافقت هذا التطور في الحاسبات تطور في أنظمة التشغيل و طرق المعالجة. فظهرت أنظمة التشغيل المتسلسلة، أنظمة تشغيل متعددة البرمجيات، و أخرى ذات وقت و متعددة المعالجات التنفيذ مشترك. و بالرغم من أن الخوارزميات عادة لا تؤخذ بعين الاعتبار في أجيال الحاسبات ، إلا أنها أيضاً تتطور و بثبات ، و يشمل ذلك الخوارزميات المستخدمة في العلوم الحاسوبية. [39]

الجيل الأول للحاسبات الآلية(1938–1953):

اعتمد الحاسب في الجيل الأول على الأنابيب المفرغة و أيضا أسستخدمت لغة الآلة و التي تتكون من رقمين فقط هما (0 و 1) في برمجته و كذلك الشريط الممغنط كوحدة تخزينية سريعة و ذات طاقة عالية مع قارىء البطاقات المثقبة كوحدة إدخال و إخراج للحاسب.

- قدرة المعالجة لتلك الحاسبات المبكرة تقدر بعشرة آلاف تعليمة كل ثانية.
- أما عن التخزين فكان بإمكان تلك الحاسبات تخزين 2000 حرف أبجدي أو رقمي.
- ظهر في الجيل آلة "فون نيومان" IAS كانت الأولى في توظيف علم الحساب المتوازي و التي بدأ "فون نيومان" بتصميمها في عام 1946 ، و لم تكتمل بجميع وظائفها إلا في عام 1952.
- كان الحاسب الآلي في هذا الجيل تسلسلي العمارة ، و فيه طرحت اقتراحات لآلات متوازية و بدرجات متفاوتة ، و لكنها لم تكن تتجاوز مرحلة التنميط (النمذجة).
- كانت البرامج في أول الجيل تكتب بلغة الآلة ، و مع بدء الخمسينات بدأ استخدام لغة التجميع و كانت الترجمة من لغة التجميع إلى لغة الآلة تتم يدويا ، و بعد ذلك تم عمل المجمع Assembler و الذي يقوم بعملية تحويل البرامج من لغة التجميع إلى لغة الآلة.
  - من الحاسبات المشهورة في الجيل الأولENIAC وEDVAC لمؤسسة Bell.

[39] :M.Eleuldj, Département Génie Informatique, EMI, septembre 2014, p7-12

الجيل الثاني(1952–1963): في هذا الجيل حدثت تطورات هامة جداً على كل المستويات من بناء الدارات الأساسية إلى لغات البرمجة ، و من المميزاته ما يلي:[40]

- استُخدمت في هذا الجيل الترانزستورات بدلاً من الصمامات المفرغة.
- قدرة المعالجة لحاسبات الجيل الثاني تقدر بمائتي ألف تعليمة كل ثانية.
- يمكن لحاسبات هذا الجيل تخزين 32000 حرف أبجدي أو رقمي.
- تم في هذا الجيل إنتاج العديد من لغات البرمجة العالية المستوى ، مثل (FORTRAN 1956) ، و (ALGOL 1958) ، و (COBOL 1958).
- تنفيذ البرامج يتم بالتسلسل.
- من الآلات الهامة في الجيل الثاني TRADIC, IBM 1620 و الآلة IBM 704 و الي وصلت سرعتها حوالي خمسة ألاف عملية حسابية في الثانية (5 KFLOPS).

الجيل الثالث (1962–1975): ظهرت في هذا الجيل الدوائر المتكاملة و هي عبارة عن دوائر إلكترونية جمعت في شريحة صغيرة و تحتوي على ملايين من المعدات الإلكترونية.

- ظهرت في هذا الجيل الحاسبات IBM360/91 ، TI–ASC ، Cyber–175 ، STAR–100 ، و الحاسب Illiac IV عام (1964) الذي يحتوي على ستة عشر(16) معالج و هناك نسخة منه مكونة من أربعة و ستون (64) معالج و قد استعملت من طرف مؤسسة NASA الأمريكية سنة 1972. و اقترح في سنة 1966 الباحث Flyn تصنيف لبنية الحاسبات. [41]
- ظهرت مترجمات لغات البرمجة الذكية.
- ظهور البرمجة المتعددة (تنفيذ عدد من برامج في نفس الوقت).
- ظهور نظام المشاركة في الوقت (Time sharing)، و هي عملية تنظيم مهام الحاسب المختلفة من عمليات إدخال و إخراج ومعالجة للوصول إلى الأستخدام الأمثل لوحدة المعالجة المركزية، مما يساعد على سرعة استجابة الحاسب، و يشعر كل مستخدم بأنه الوحيد الذي يتعامل مع الحاسب مع وجود عدد كبير من المستخدمين.[42]

---

[40] M. Eleuldj: ، مصدر سبق ذكره ، ص9

[41] :M. DALMAU, "Les superordinateurs", IUT de BAYONNE, p.2

[42] M. Eleuldj: ، مصدر سبق ذكره ، ص9

الجيل الرابع(1972–1990): نلخص أهم معالم هذا الجيل في الآتي:[43]

- استُخدمت في هذا الجيل الدوائر المتكاملة الواسعة LSI(Large scale integration) كما استخدم في هذا الجيل المعالج الدقيق (Micro–processor). كما تطورت وسائل تخزين البيانات كأقرص الليزر، و الأقراص و الأشرطة الممغنطة.
- من الحاسبات الهامة في هذا الجيل:
  - بالنسبة للحاسبات الشعاعية نذكر: CRAY–1 سنة 1976 و هو أول حاسب شعاعي مكون من معالج واحد تصل سرعته إلى (133 MFLOPS)، Cyber–205 سنة 1982.
  - بالنسبة للأنظمة متعددة المعالجات نذكر: IBM370/168MP و IBM3081 في سنة 1980 ، Cray X–MP سنة 1982 و 1984 و هذا الأخير متكون من إثنين ثم إلى نماذج من أربع معالجات تعمل بالتوازي و تصل سرعتها إلى (500 MFLOPS).[44]

الجيل الخامس (1990 و حتى الآن): يتميز هذا الجيل في الآتي:

– ظهور حاسبات ذات معالج أو معالجات متعددة النواه(Processor multicores)

– ذاكرة مخبأة من عدة مستويات(من 3 إلى 7 مستويات)

– المعالجة المتعددة (Multithreading)

من بين حاسبات هذا الجيل: CRAY–2 و الذي يحتوي على أربع معالجات و له سرعة تنفيذ تفوق مليار عملية في الثانية (1GFlops).

و يمكن تصنيف حواسيب المستقبل في ثلاث اتجاهات أو محاور. أول هذه الاتجاهات يختص الجانب المادي (Hardware) ، و ثاني هذه الاتجاهات هو طرق **العمل على التوازي** و الاتصالات ، و ثالث هذه الاتجاهات يختص بالبرمجيات (Software).

ففي الجانب المادي يتزايد عدد المعالجات و سرعتها. و أما المكونات المادية الأخرى مثل الذاكرة، تتزايد أحجام و سعة هذه الأخيرة في الرقاقة الواحدة.

[43] M. Eleuldj: ، مصدر سبق ذكره، صص9-10

[44] M. DALMAU:، مصدر سبق ذكره ، ص3

و في مجال أساليب **العمل على التوازي** و الاتصالات فقد تبين إمكانية تنفيذ ملايين التعليمات في الثانية الواحدة و ذلك عن طريق استخدام أكثر من معالج. و تبين أن تعاون المعالجات في تنفيذ التعليمات يكون أيسر إذا كانت تعليمات هذه المعالجات بسيطة. والجدول (1-4) يبين أهم خصائص نماذج الحواسيب الأكثر تمثيلا للفترة الممتدة من 1985 إلى غاية 1996.

الجدول(1-4):نماذج لحاسبات متوازية للفترة (1985-1996)

السنة	الصانع	النموذج	الساعة(Mhz)	عدد المعالجات	نوع المعالج	سرعة المعالج(Gflops)	سعة الذاكرة(GO)
1985	Cray	Cray2	245	1 إلى4	vectoriel	1.951	4
1988	Cray	YMP	166	2 إلى8	Vectoriel	2.667	2
1989	nCUBE	NCUBE2	20	8إلى8192	Maison	19.7	256
1990	NEC	SX-3	400	1 إلى4	Vectoriel	25.6	8
1991	INTEL	Paragon XP/S	50	64 إلى6768	Intel i860	338	128
1991	CRAY	YMP C90	245	2 إلى 16	Vetoriel	15.6	16
1992	Thinking machine	CM-5	32	16 إلى16384	Vectoriel	2028	32
1992	Meiko	Computing surface 2	50	8إلى 1024	1SPARC&2vectoriel	204.8	128
1993	Cray	T3D	150	32 إلى2048	DEC ALPHA 21064	307.2	128
1993	Fujitsu	VPP500	100	7 إلى222	Vectoriel	355	56,8
1993	Cray	Cray3	475	1 إلى16	Vectoriel	15.17	4
1994	IBM	SP2	66	8 إلى128	Power 2	34.1	32
1995	NCUBE	NCUBE3	250	8 إلى65536	Maison	6500	1لكل معالج
1995	SGI	Power Challenge	90	2 إلى128	MIPS R8000	46	16
1995	NEC	SX-4	125	1 إلى512	Vectoriel	1024	128
1995	Cray	YMP T90	450	2 إلى32	vectoriel	58.2	8
1996	SGI(cray)	T3E	300	1 إلى2048	DEC ALPHA 21164	1229	2 لكل معالج
1996	Hitachi	SR 2201	150	32 إلى2048	RISC	614.4	1 لكل معالج
1996	Fujitsu	VPP700	154	8 إلى256	CMOS VECTORIEL	614.4	512

المصدر:M.DALMAU ، المصدر سبق ذكره، ص.6

نلاحظ من خلال الجدول(1-4) أن الحاسبات في تطور مستمر و هذا سواء على مستوى سعة الذاكرة ، عدد المعالجات ، عدد النواة المكونة للمعالج و سرعة المعالج في تنفيذ العمليات. فعلى سبيل المثال في سنة 1985 كان الحاسب Gray2، سعة الذاكرته لا تتعدى (6 GO) ، سرعة المعالج 1.951 (Gflops) و عدد المعالجات أربعة (4) بينما في سنة 1996 و بالنسبة للحاسوب CMOS فسعة الذاكرة أصبحت (512 GO) و سرعة المعالج تساوي (614.4 Gflops) و عدد المعالجات يقارب 256. و لقد شهدت الحاسبات بعد هذه الفترة تطورا كبيرا سواء في سرعة أداءها أو عدد المعالجات المكونة لها، نلخص أفضل هذه الأنظمة من حيث سرعة الأداء و بالترتيب لشهر نوفمبر 2018 في الجدول(1-4-أ) الموالي:

الجدول(1-4-أ): نماذج لأفضل أنظمة الحاسبات لسنة 2018

النظام	الصانع/البلد	عدد النوى(cores)	الأداء Rmax(TFlops)	الذروة النظرية RPeak(TFlops)	القوة(KWatt)
Summit	IBM/USA	2397824	143500.0	200794.9	9783
Sierra	IBM/USA	1572480	94640.0	125712.0	7438
Sunway	NRCPC/China	10649600	93014.6	125435.9	15371
Tianke-2A	Nudt/Chine	4981760	61444.5	100678.7	18482
Piz Daint	Cray Inc/Switzeland	387872	21230.5	41461.2	7578

المصدر: https://www.top500.org

نلاحظ من خلال الجدول(1-4-أ) أن عدد المعالجات المكونة للنظام (الحاسب العملاق) قد أرتفع كثيرا إذ يفوق مليونين في الحاسبSummit لصانعه IBM و أنه يمكنه أداء حوالي 200 TFlops عملية حقيقية في الثانية وهو أسرع بكثير من أسلافه في الجدول(1-4) والتي كانت سرعتها لا تتعدى إنجاز (614.4 Gflops) عملية في الثانية.

خلاصة:

تعرضنا في هذا الفصل إلى تقديم مفاهيم عامة حول مختلف طرق الحساب أو المعالجة في الحاسوب (الحساب المتسلسل ، الحساب المتوازي و الحساب الموزع). وضحنا ذلك من خلال أشكال لبنية الحاسوب. تطرقنا كذلك إلى الفائدة من استخدام التوازي و تعدد المعالجات و كذلك ذكر بعض ميادين تطبيق المعالجة المتوازية. تطرقنا أيضا إلى دراسة مدى نجاعة الخوارزميات المتوازية نظريا في تسريع الحساب وحساب تكلفة العمل المتوازي و إلى مختلف أشكال معالجة المعطيات على التوازي (على مستوى البرنامج ، على مستوى الإجرائية و على مستوى التعليمة) و وضحنا ذلك من خلال أشكال و أمثلة.

في الأخير، سردنا موجز لتاريخ الحاسبات و خصائصها حسب كل جيل منها.

الفصل الثاني

تصنيف الحاسبات و الخوارزميات المتوازية

تمهيد :

ليس هناك ، في الوقت الحالي' تصنيف وحيد و عالمي لبنية الحاسبات سواء كانت وحيدة أو متعددة المعالجات مرتبطة (أو لا) بشبكة معلوماتية. أنواع الآلات الحالية كثيرة جدا ، لذا قام الباحثون بأعمال بحثية هامة بغية تحقيق تصنيف لها. أقدم تصنيف و الأكثر استعمالا هو تصنيف فلان [Flynn] الذي يرتكز على تحليل تدفقات التحكم و تدفقات البيانات. و يعتمد تنظيم البنى المتوازية على الموارد التي تكون البنى التسلسلية كوحدة المعالجة ، وحدة التحكم ، الذاكرة و وحدات الإدخال و الإخراج (القرص الصلب، الشبكات ، ..إلخ). و أثناء التنفيذ تتبادل جميع هذه الوحدات المعلومات بواسطة مورد إضافي و هو شبكة التواصل الداخلي[1].

يتم تصنيف الحواسيب تبعا لطريقة تنفيذ التعليمات على البيانات وتتم برمجة كل نوع من أنواع الحواسيب باستخدام خوارزمية إما تسلسلية و إما متوازية كل على حسب ما يوفر من إمكانيات. كما أنه من الضروري وجود اتصال بين وحدات المعالجة بعضها مع بعض خلال الحساب و ذلك من أجل تغيير البيانات و يتحقق هذا الاتصال عن طريق الذاكرة المشتركة للنظام أو عن طريق شبكة الاتصالات بين وحدات المعالجة.

1.2 تصنيف فلاين للحاسبات[Flynn's Classification].

يعتبر تصنيف فلاين للحاسبات من أقدم التصنيفات و أكثره استعمالا. و ترتكز هذه التصنيفات على تحليل تدفق المراقبة (تدفق التعليمات) و تدفق البيانات سواء كانت وحيدة أو متعددة و توصل إلى أربع نماذج كما هو موضح في الشكل(2−1)[2]. و نعني بالتدفق ذلك التتابع أو التسلسل للتعليمات أو المعطيات كما نفذت بواسطة المعالج. فعلى سبيل المثال، تقوم بعض الآلات بتنفيذ تدفق واحد ، بينما يتم تنفيذ عدة تدفقات بالتوازي في آلات أخرى. و بنفس الطريقة فبعض الآلات تعالج تدفق واحداً من المعطيات ، و آلات أخرى تعالج تدفقات متعددة[3].

[1] : N. Hameurlain, "Architectures Parallèle", p.11

[2] :Langages de description d'architectures matérielles hybrides

[3]: Daniel C. Hyde, "Introduction to the Principles of parallel Computation", Bucknell University, p.16

شكل (2-1): تصنيف فلاين للحاسبات

	Data Streams (دفق معطيات)		
Instruction Stream (دفق تعليمات)		Single(وحيدة)	Multiple(متعددة)
	Single	SISD	SIMD
	Multiple	MISD	MIMD

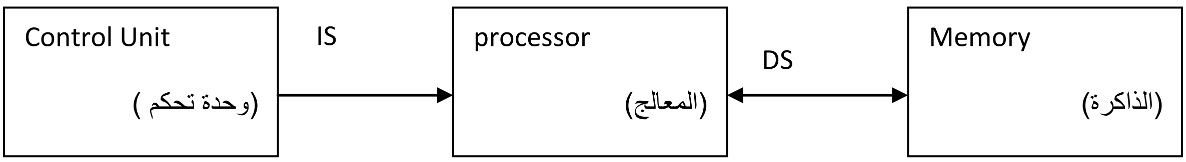
المصدر: DANIEL C. Hyde، ص.30 ، ص16

1.1.2 نموذج الحاسبات وحيدة تدفق التعليمات و وحيدة تدفق المعطيات SISD: Single Instruction stream, Single Data stream(SISD)

Single Instruction stream, Single Data stream(SISD) : هذا النموذج يوافق الحاسب التسلسلي لفون نيومان Von Neumann الذي صممه في أواخر الأربعينات و أوائل الخمسينات من القرن الماضي، و تتميز هذه الحاسبات على أنها ذات معالج وحيد و الذي يقوم بمعالجة سلسلة معطيات الواحدة تلوى الأخرى[4]. يدخل في هذا الصنف جميع الحاسبات الكلاسكية التسلسلية التقليدية . تُحضر الحاسبات من هذا الصنف التعليمات من الذاكرة ثم تقوم بتنفيذها عادة باستخدام قيم المعطيات المشار إليها من الذاكرة. و من ثم تقوم بإحضار تعليمات أخرى من الذاكرة ، و هكذا. تنفذ تعليمة واحدة فقط في كل لحظة t بالتتابع[5].

هناك العديد من المراجع التي تناولت تصنيف فلاين للحاسبات و كمثال لحاسبات SISD نذكر VAX11/780 و IBM370/168[6].

الشكل (2-2): تصميم حاسبات SISD



المصدر 1: Parallel Algorithm Quick guide, p.2

حيث: IS (Instruction Stream) هي تدفق التعليمات.

DS (Data Stream)هي تدفق المعطيات

[4] : Langages de description d'architectures matérielles hybrides

[5]: Daniel C. Hyde ، مصدر سبق ذكره ، ص.16

[6] :M. Eleuldj, "Architectures parallèles", Département Génie Informatique, EMI,2014, p19

مثال (2-1): برنامج SISD [7]

PU (processor Unit)	time
Load A	
Load B	
C= A+ B	
Store C	
A= A*2	
Store A	

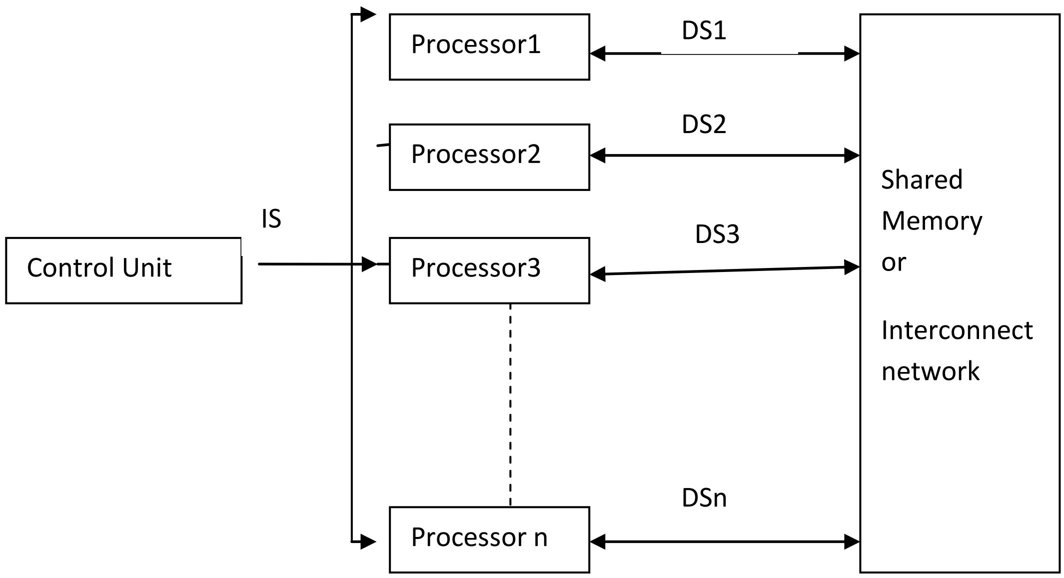
2.1.2 نموذج الحاسبات وحيدة تدفق التعليمات و متعددة تدفق المعطيات SIMD:

Single Instruction stream, Multiple Data stream(SIMD)

يشبه هذا النموذج في تصميمه نموذج SISD لأنه يحتوي على وحدة تحكم واحدة و لكن عدة معالجات حسابية متجانسة والتي تقوم بأداء نفس العملية الصادرة من وحدة التحكم ولكن على بيانات مختلفة و بصفة تزامنية[8].و بما أن كل عنصر معالجة يستعمل معطياته الخاصة هذا يعني وجود عدة تدفقات للبيانات. يتم التواصل بين المعالجات في الحاسبات SIMD بواسطة الذاكرة المشتركة أو من خلال شبكة الربط [9]. يلائم هذا النوع من الحاسبات، العمليات على الأشعة و على المصفوفات ، و غالبا ما يستخدم من أجل عمليات الحساب العلمي، تتطلب معالجات مختصة، كما توافق هذه الأجهزة المعالج الرسومي[10](GPU) .

[7] : Amara Yacine , Le GPU ;un moyen pour le calcul intensif , Ecole Militaire polytechnique, Alger , P.13
[8] : Langages de description d'architectures matérielles hybrides
[9]: Parallel Algorithm - Quick Guide ، مصدر سبق ذكره ، ص.2
[10] : "Architectures parallèles", support de cours, département de genie électrique et genie informatique, Université Laval, p.11.

الشكل (2-3): تصميم حاسباتSIMD ذات ذاكرة مشتركة.



المصدر: Parallel Algorithm Quick guide, p.2

من الآلات الهامة التي تتبع لتصنيف SIMD هي: ILLIAC IV و MPP [11] .

مثال(2-2): برنامج SIMD.

هذا مثال لبرامج يمكن تنفيذها على حاسوب من نوع SIMD. و الذي يحوي نفس التعليمات و لكن معطيات مختلفة[12].

time

P_1		P_2		P_N
Prev instruct		Prev instruct		Prev instruct
Load A(1)		Load A(2)		Load A(n)
Load B(1)		Load B(2)		Load B(n)
C(1)= A(1)*B(1)		C(2)=A(2)*B(2)		C(n)=A(n)*B(n)
Store C(1)		Store C(2)		Store C(n)
Next instruct		Next instruct		Next instruct

[11]: M. Eleuldj، مصدر سبق ذكره ، ص20

[12] :Amara Yacine, "Le GPU: un moyen pour le calcul intensif", Ecole Militaire Polytechnique, Alger, p.13

من ناحية أخرى يمكن أن نقسم فيئة الحواسيب هذه إلى أربع فئات فرعية وفقا لقدرة إثنين أو أكثر من المعالجات على الوصول إلى موقع الذاكرة نفسه و بنفس الوقت و ذلك كما يلي[13]:

أ- حاسبات Exclusive-Read,Exclusive-Write(EREW)SM SIMD

في هذا النوع من الحاسبات لا يسمح لمعالجين بالقراءة و الكتابة في نفس الوقت من الموقع نفسه من الذاكرة.

ب- حاسبات Concurrent-Read, Exclusive-Write (CREW) SM SIMD

في هذا النوع من الحواسب يستطيع معالجين أو أكثر القراءة من نفس موقع الذاكرة في نفس الوقت ، و لكن بالمقابل لا يسمح لهما بالكتابة في نفس الموقع من الذاكرة في نفس الوقت. بمعنى يسمح لمعالج واحد بالكتابة فقط .

ج- حاسبات Exclusive-Read, Concurrent-Write (ERCW) SM SIMD

يمكن في هذا النوع من الحواسب بالكتابة في نفس موقع الذاكرة من طرف معالجين ولكن لا يسمح لهما بالقراءة . أي أنه لا يمكن لمعالجين القراءة من نفس موقع الذاكرة في الوقت نفسه.

د- حاسبات Concurrent-Read, Concurrent-Write (CRCW) SM SIMD

في هذا النوع من الحواسيب تتحقق ميزة القراءة المتعددة و الكتابة المتعددة.

إن خاصية القراءة المتعددة و الوصول إلى نفس الموقع من الذاكرة لن يؤدي إلى أية مشاكل حيث أن كل المعالجات التي تقرأ من ذلك الموقع تصنع نسخة من محتويات الموقع و تخزنها في ذاكرتها المحلية الخاصة و لكن المصاعب تظهر عند الوصول للكتابة المتعددة. فإذا حاولت عدة معالجات أن تخزن بيانات مختلفة في عنوان محدد بنفس الوقت فأي منها سينجح؟ و بمعنى آخر حتما سيكون هناك طريقة لتحديد محتويات ذلك العنوان بعد عملية الكتابة[14].

[13]: كندة زين العابدين، " **خوارزميات المعالجة المتوازية و برمجتها**"، جامعة دمشق، 2006 ، ص.17.

[14]: كندة زين العابدين ، مصدر سبق ذكره ، ص17

إن السماح بالقراءة المتعددة لا تشكل عائق في البرنامج، بينما الكتابة المتعددة في نفس الموقع من الذاكرة يتطلب برامج أو بروتوكلات تحكيم، أبرزها ما يلي[15]:

- بروتوكل المشترك، الذي يسمح بالكتابة المتعددة عندما تكون جميع القيم المراد كتابتها متساوية.
- بروتوكول (arbitrary)، و الذي يسمح لمعالج بعملية الكتابة و لا يسمح للمعالجات الأخرى.
- بروتوكل الأولوية، و فيه جميع المعالجات منظمة في قائمة حسب الأولوية، و يسمح للمعالج بأعلى أولوية بعملية الكتابة بينما المعالجات لا يسمح لها بذلك.

3.1.2 نموذج الحاسبات متعددة تدفق التعليمات و وحيدة تدفق المعطيات MISD:

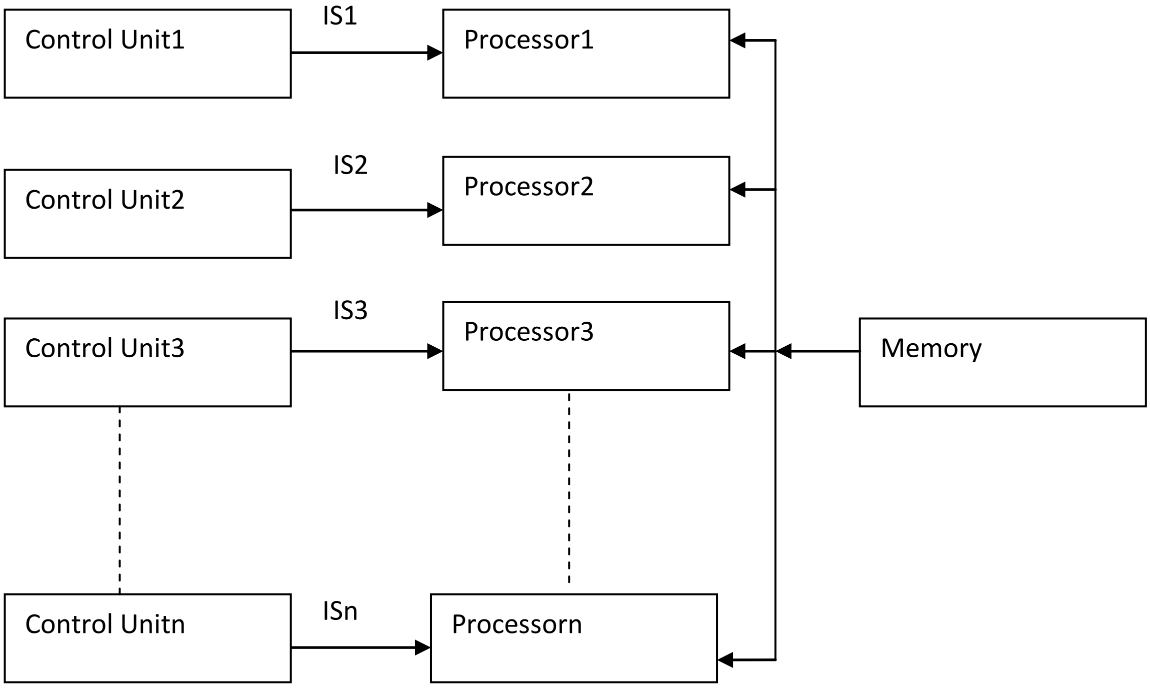
(Multiple Instruction stream, Single Data stream)

:يتم تقسيم وحدات المعالجة و وحدات التحكم إلى مستويات. كل عملية تسند لمستوى معين و تنفذ على نفس المعلم و في نفس الوقت. أو بعبارة أخرى يتم في هذا النوع تنفيذ عدة تعليمات مختلفة على معامل واحد(Single Data) خلال الدورة الزمنية للحاسب. يوجد القليل من الحاسبات المتوازية من نوع MISD. يطلق على هذا النوع من الحاسبات بنموذج "الأنابيب" و أن MISD هي قليلة الاستعمال فهي تعتمد مبدأ العمل التسلسلي[16] .

[15] : Vipin Kumar, George Karypus,......, "Introduction to parallel Computing", Addison Wesly, Second Edition,2003, p.57

[16] : langages de description d'architectures materielles hybrides ، ص25

الشكل (2-4): تصميم حاسبات MISD



المصدر: Parallel Algorithm Quick guide, p.3

مثال(2-3): برامج MISD: يتم في المثال(2-3) تنفيذ عدة برامج (تعليمات متعددة) على تدفق وحيد للمعطيات.[17]

P1		P2		Pn	time
Prev instruct		**Prev instruct**		**Prev instruct**	↓
Load A(1)		**Load A(1)**		**Load A(1)**	
C(1)= A(1)*1		**C(2)=A(1)*2**		**C(n)= A(1)*n**	
Store C(1)		**Store C(2)**		**Store C(n)**	
Next instruct		**Next instruct**		**Next instruct**	

[17] Amara Yacine: ، مصدر سبق ذكره ، ص13

4.1.2 الحاسبات متعددة تدفق التعليمات و متعددة تدفق المعطيات MIMD:

لحاسبات MIMD، عدة وحدات التحكم، و عدة وحدات معالجة و ذاكرة مشتركة أو شبكة ربط. ويكون التحكم في هذا النوع موزع بين جميع المعالجات و التواصل يتم عن طريق تبادل الرسائل، و كل معالج يمكنه تنفيذ خوارزمي مختلف على بيانات مختلفة (أو لا)[18]. و يوافق هذا النوع الحاسبات متعددة المعالجات ذات وحدات تحكم مستقلة بعضها عن بعض والتي تقوم كل واحدة منها باصدار أوامرها لوحدة المعالجة الخاصة بها لتنفيذ عمليات على بيانات مختلفة (أو لا) و بطريقة غير متزامنة[19].

ينقسم هذا الصنف من الحاسبات على قسمين و هما حاسبات MIMD ذاكرة مشتركة و حاسبات MIMD تمرير الرسائل.من بين الحاسبات التي تنتمي لهذا الصنف نذكر على سبيل المثال:

(IBM 360/16 MP, Cray-2, IBM 3081/30845)[20].

و يوجد في هذا التصنيف فئتين فرعيتين هامتين و هما:

(a) الذاكرة المشتركة (Shared memory).

(b) تمرير الرسائل (message passing) أو شبكة اتصالات

2. 1. 4 a نموذج الحاسبات المتوازية ذات الذاكرة المشتركة:

فالحاسبات المتوازية ذات الذاكرة المشتركة تعرف بمتعددة المعالجات (multiprocessors)، بينما تلك التي تستعمل شبكات الربط تسمى بمتعددة الحواسيب(multicomputers). كما يوجد كذلك نوعان لهذه الأخيرة مصنفة حسب المسافة بين المعالجات. هذه الحاسبات تتميز بساعة داخلية مستقلة لكل معالج، و لكن ذاكرة وحيدة مشتركة بين المعالجات، أين يمكنها القراءة و الكتابة في نفس المجال من الذاكرة مما يمكن من تحقيق التوازي في المعطيات و في التعليمات. هنا المبرمج ليس له دخل في تحديد

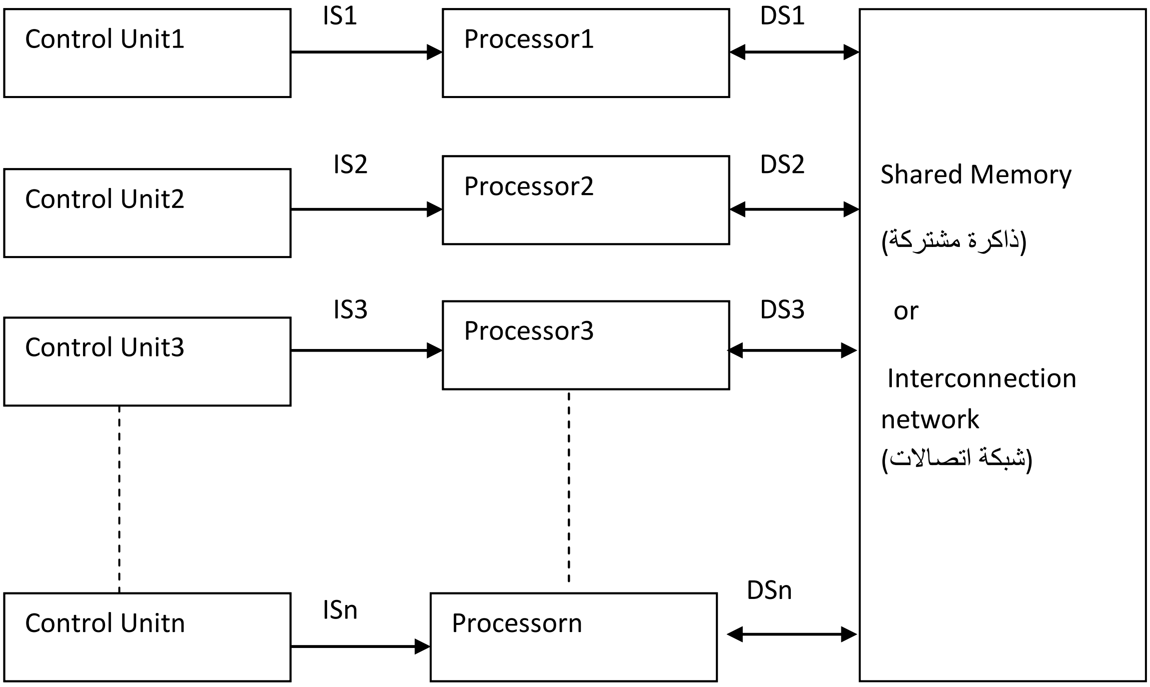
[18] :Rédha LOUCIF, "parallélisation d'Algorithmes d'Optimisation Combinatoire", 2014, p22

[19] : langage de description d'architectures materielle hybrides, p25

[20]:M. Eleuldj ، مصدر سبق ذكره ، ص22

أماكن تخزين المعطيات في الذاكرة' فقط تحديد الجزء من البرنامج الذي يتم تنفيذه من طرف هذا المعالج أو ذلك بالإضافة إلى إدارة التزامن بين المعالجات[21].

الشكل (2-5): تصميم حاسبات MIMD (ذاكرة مشتركة)



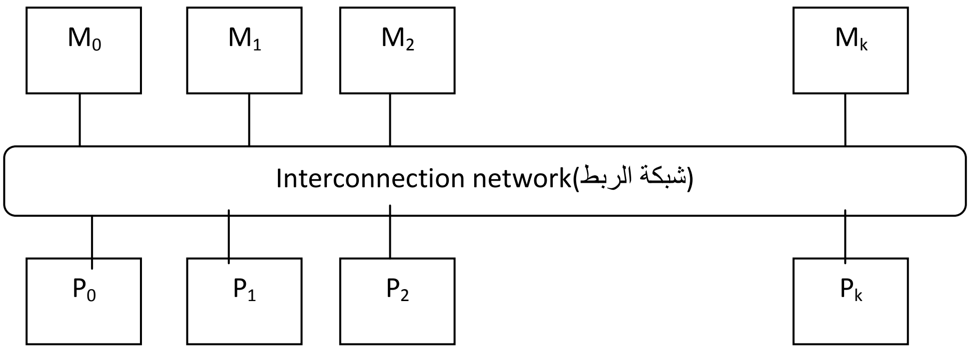
المصدر: Parallel Algorithm Quick guide, p.3

[21] Rédha LOUCIF: ، مصدر سبق ذكره ، ص32

مثال(2−4): برامج MIMD :يحتوي هذا المثال على عدة برامج p1, p2 ,....pn مستقلة يمكن تنفيذها في نفس الوقت و على معطيات مختلفة.[22]

P1		P2		Pn	time
Prev instruct		Prev instruct		Prev instruct	↓
Load A(1)		Call funcD		Do 10 i=1,N	
Load B(1)		X=y*z		Alpha= w**3	
C(1)= A(1)*B(1)		Sum=x*2		Zeta= C(i)	
Store C(1)		Call sub1(i,j)		10 continue	
Next instruct		Next instruct		Next instruct	

الشكل (2−6): نموذج الذاكرة المشتركة(MIMD Shared Memory)



المصدر : Daniel C .Hyde ، مصدر سبق ذكره ،ص. 19

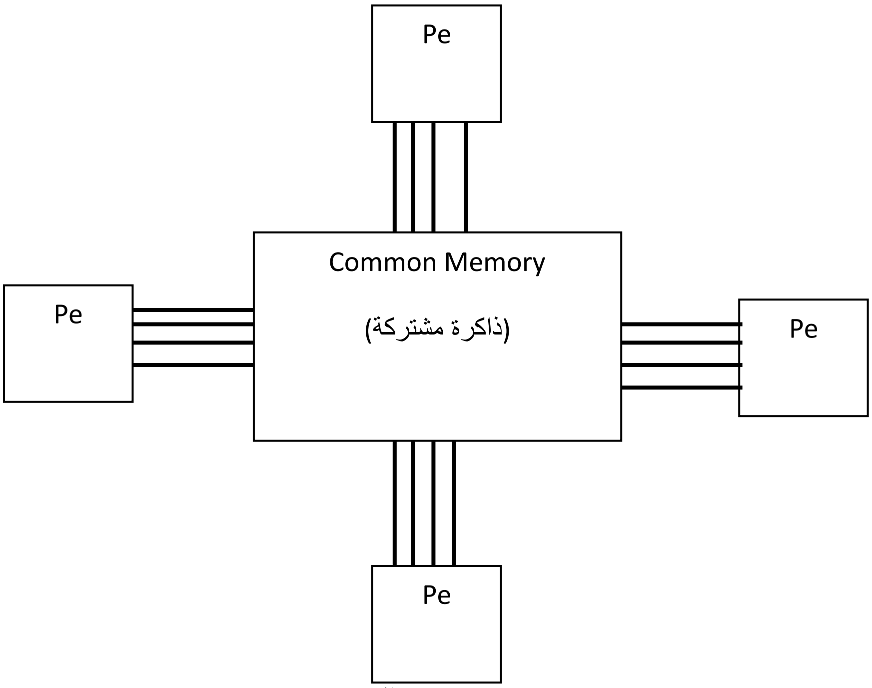
في هذا النموذج، المعالجات (p_i) متصلة بوحدات الذاكرة بواسطة شبكة الربط، و التي يمكنها أخذ عدة أشكال مرتبطة بالآلة. قد تكون شبكة الربط حلقية ، مصفوفية أو غيرها. يكون زمن الولوج للذاكرة

[22] Amara Yacine، مصدر سبق ذكره ، ص13

عبر شبكة الربط هو المحدد لنجاعة الآلة. نقدم في ما يلي ثلاث نماذج لحاسبات متوازية ذات الذاكرة المشتركة موجودة بالسوق لتوضيح مختلف البنى و بشكل خاص شبكات الإتصال بها [23].

النموذج الأول: الحاسب Cray X–MP: يتكون هذا الحاسب من أربع عناصر معالجة (Pe) و لكل عنصر معالة أربع منافذ (ports) للربط بالذاكرة المشتركة و هو ما يوضحه الشكل(2–7).

الشكل (2–7): الحاسب Cray X–MP



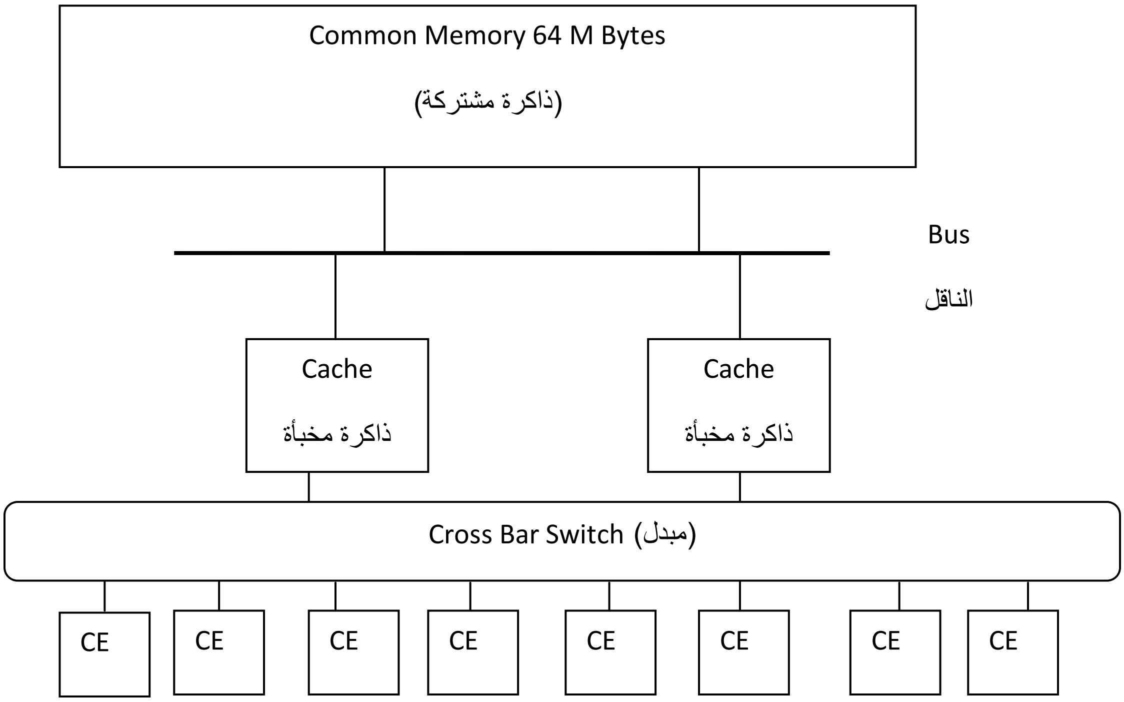
المصدر : Daniel C .Hyde ، مصدر سبق ذكره ،ص. 19

النموذج الثاني: الحاسب فائق السرعة The Alliant FX/8 : يمكن للحاسب "Alliant FX/8" احتواء حتى ثمانية عناصر حسابية (CEs)، و تتقاسم هذه العناصر الحسابية فيما بينها ذاكرة مشتركة. تتصل هذه العناصر بذاكرتين مخبأتين (cache) من 64 كيلو بايت (64kbyte) لكل منها و ترتبط عبر شبكة تبديل العارضة Crossbar Switch. و ترتبط الذاكرتان المخبأتان بالذاكرة المشتركة من خلال ناقل ذو سعة 188 ميغا بايت/ ثانية[24].

[23]: Daniel C .Hyde ، مصدر سبق ذكره، ص19

[24]: Daniel C .Hyde ، مصدر سبق ذكره، ص20

الشكل (2-8): الحاسب Alliant FX/8 بثمانية معالجات (CEs) تشترك في الذاكرة

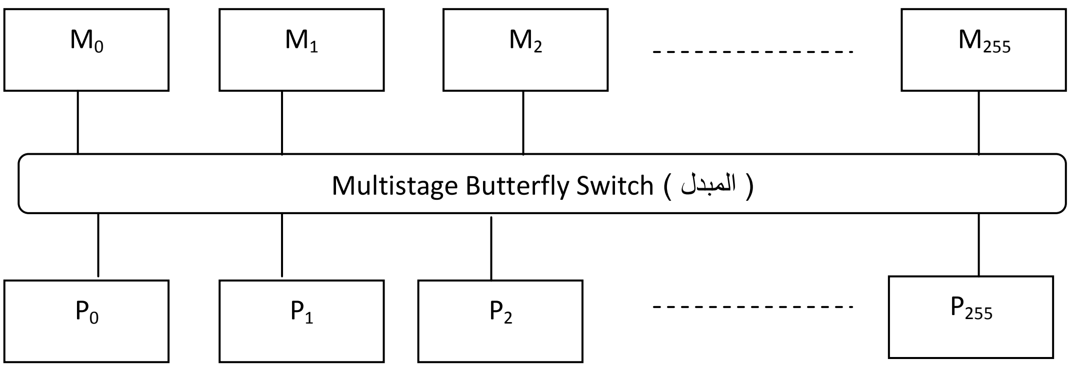


المصدر: Daniel C .Hyde, ، مرجع سبق ذكره، ص. 20

النموذج الثالث: الحاسب The Bolt , Beranek and Newman (BBN) Butterfly

يتكون هذا الحاسب من 256 معالج متصلة بذاكرة موزعة عبر شبكة الربط الفراشة (Multistage Butterfly Switch)، حيث يمكن فيه لأي عنصر معالجة P_i أن الولوج إلى أي وحدة من الذاكرة . بمعنى يستعمل هذا الحاسب فضاء العناوين المشتركة للذاكرة[25].

الشكل (2-9): الجهاز BBN Butterfly ذاكرة M_i (i=1,.....255).



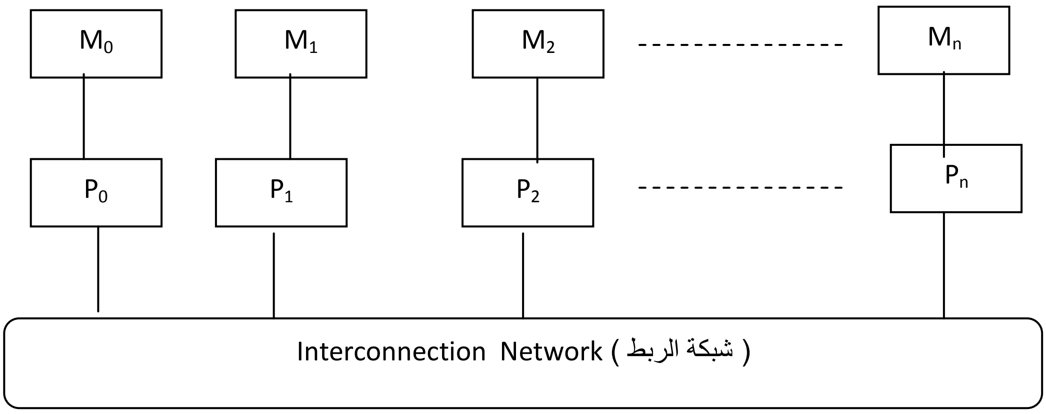
المصدر: Daniel C .Hyde, ، مرجع سبق ذكره ص. 20

[25]: Daniel C .Hyde، مصدر سبق ذكره، ص20

b. 4.1.2 الحاسبات المتوازية بتمرير الرسائل MIMD Message Passing

في نموذج تمرير الرسائل ، جميع المعالجات لها ذاكرتها الداخلية الخاصة بكل منها. و للتواصل بينها، تقوم المعالجات بإرسال رسائل لبعضها البعض عبر شبكة الربط . و يمكن لهذه الأخيرة أن تأخذ أشكالا مختلفة. فشبكة الربط المعروفة بكثرة في هذا النموذج هي " المكعب الثنائي متعدد الأبعاد" k–dimentional . فعلى سبيل المثال في المكعب ثلاثي الأبعاد ، توضع المعالجات في زواياه كما هو موضح في الشكل (2–11)[26] . الميزة الأساسية في هذا النموذج هي سهولة زيادة عدد المعالجات فيه بوسائل بسيطة مثل(CLUSTERS) و لكن من جهة أخرى صعوبة كبيرة في برمجته[27].

الشكل (2–10): نموذج MIMD تمرير الرسائل

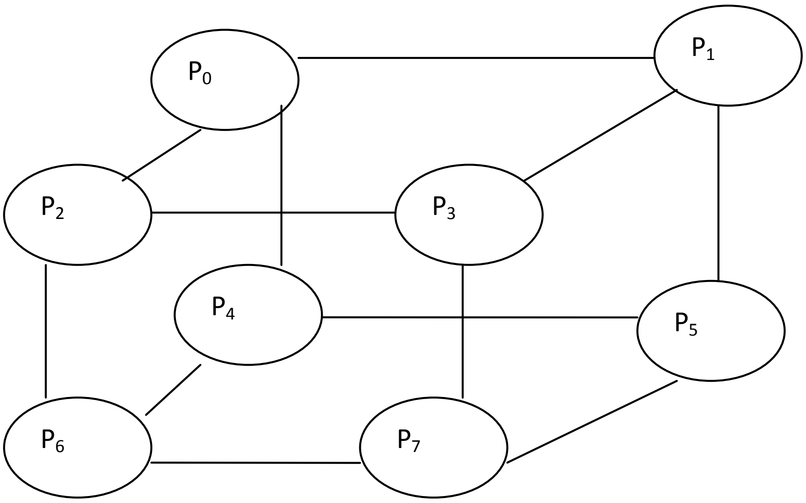


المصدر: Daniel C .Hyde, ، مصدر سبق ذكره ص 21

[26]: Daniel C .Hyde ، مصدر سبق ذكره ، ص21

[27]: Rédha Loucif ، مصدر سبق ذكره ، ص33

الشكل (2-11): مكعب ثلاثي الأبعاد (المعالجات P_i توضع على زواياه)



المصدر: Daniel C .Hyde, ، مصدر سبق ذكره ص 21

ظهرت تصنيفات أخرى بعد بعد فلاين و هي في الحقيقة تعتبر أمتدادات لها، تمحورت على عدد المعالجات، الولوج للأقراص و إلى الذاكرة المستعملة[28].

2 .2 تصنيفات أخرى بديلة للحاسبات المتوازية:

أقترح الباحث [Stonebraker 86] تقسيم بنية الحاسبات متعددة المعالجات إلى أربع أقسام:

- في القسم الأول كل شيىء مشترك (Shared-Everything):يمكن لكل معالج الولوج لكل الأقراص و كل مواقع الذاكرة.
- القسم الثاني لا شيء مشترك (Shared-Nothing): ليس هناك اشتراك في الموارد، و كل معالج له أقراص و ذاكرة خاصة به.
- القسم الثالث، الأقراص مشتركة (Shared-Disks): لكل معالج ذاكرته الخاصة، و لكن الأقراص مشتركة بين المعالجات.
- القسم الرابع سمي بالهجين(Hybrid): هو نظام ذو أقراص مشتركة، بحيث تتكون عقده من أنظمة من نوع "الكل مشترك".

[28] : langage de description d'architectures matérielles hybrides, p26

تمت دراسة و مقارنة هذا التصنيف من طرف [Dewitt & al. 92] و [Bergsten & al. 93] من خلال الولوج قواعد المعطيات المتوازية و تم تحديد معايير نجاعة كل قسم.ثم تلتها تصنيفات أخرى للحاسبات المتوازية و خاصة منها ذات الذاكرة المشتركة و الذاكرة الموزعة و من روادها: [Chrichlow 88] ، [Chassin D. K. 95] و [Zomaya 96] ، و هي[29]:

- الحاسبات ذات الولوج المنتظم للذاكرة (UMA: Uniform Memory Access): تتميز هذه الفيئة بذاكرة مشتركة، المعالجات متشابهة و زمن الولوج للمعطيات المشتركة سريع و هو نفسه لجميع المعالجات ، و في الغالب تكون بنيتها منظمة حول ناقل مشترك وحيد و الذي لا يسمح بعدد كبير من المعالجات . يطلق على هذه الحاسبات SMP (Symmetric Multiprocessor).
- الحاسبات ذات الولوج الغير منتظم للذاكرة (NUMA: Non Uniform Memory Access)

تتميز هذه الفيئة من الحاسبات بزمن الولوج بغير منتظم للمعطيات، أي يختلف من موقع لآخر في الذاكرة. كل معالج له ذاكرته الخاصة و يمكنه الولوج لمواقع المعالجات الأخرى عبر شبكة الربط. تعتبر هذه الفيئة امتداد لأنظمة UMA و تجاوز مشاكل الولوج للذاكرة عبر الناقل الوحيد.

- الحاسبات ذات التوازي التامMPP (Massively Paralel Processors): و تحتوي على بعض الميئات وحتى الآلاف من المعالجات. و يدخل ضمن هذا النموذج حاسبات MIMD و حاسبات SIMD ذات الذاكرة المشتركة أو شبكة الربط .

3.2 شبكة الربط (Interconnection Network) :

تعتمد الحاسبات المتوازية على شبكات الربط في التواصل و تبادل المعطيات بين المعالجات و بين المعالجات و الذاكرات المشتركة أو الموزعة. تتألف الشبكة من وصلات ، و التي يمكن أن تكون أسلاك أو لاسلكية ،ألياف البصرية و عناصر أخرى متخصصة لنقل المعلومات، كالمبدلات(switches) و الموجهات (routers). تلعب هذه العناصر بالإضافة إلى برنامج و بروتكول توجيه الرسائل المستعمل و طوبولوجيا الربط بين المعالجات في الشبكة دورا مهما في أدائها. تختلف شبكة الربط بين المعالجات في الحاسوب المتوازي (متعدد المعالجات أو متعدد النوى) في بنيتها و استخدامتها عن الشبكات

[29] :langages de description d'architectures materielles hybrides, p26

الحاسوبية (متعددة الحاسبات). فالأولى توصل بين معالجات متقاربة و في نفس الحاسوب. بينما الثانية فتربط بين حاسبات متباعدة، قد تصل المسافة بينها إلى آلاف الكيلومترات (كشبكة الأنترنت مثلا)[30].

تعتبر شبكة الربط كذلك مهمة خاصة في الحاسبات ذات الذاكرة الموزعة لأنها تحدد سرعة الولوج لبيانات المعالج المعالج المجاور. و من خصائصها أيضا زمن بدأ التواصل (latency) ، سرعة نقل البيانات (bande passante) و طوبولوجيا الشبكة (خطية، شجرة، حلقية، مكعبية، نجمية إلى غير ذلك) [31].

يمكن تصنيف شبكات الربط إلى ثابتة و دينامكية. فالشبكة الثابتة تتمثل في التواصل نقطة–نقطة(point to point) بين عقد المعالجات و تكون بالتالي موصلة بشبكة مباشرة(direct network). و في المقابل الشبكة الدينامكية فقد بنيت باستعمال المبدلات و روابط التواصل، و بالتالي موصلة عبر شبكة غير مباشرة و هذا ما يوضحه الشكل الموالي[32].

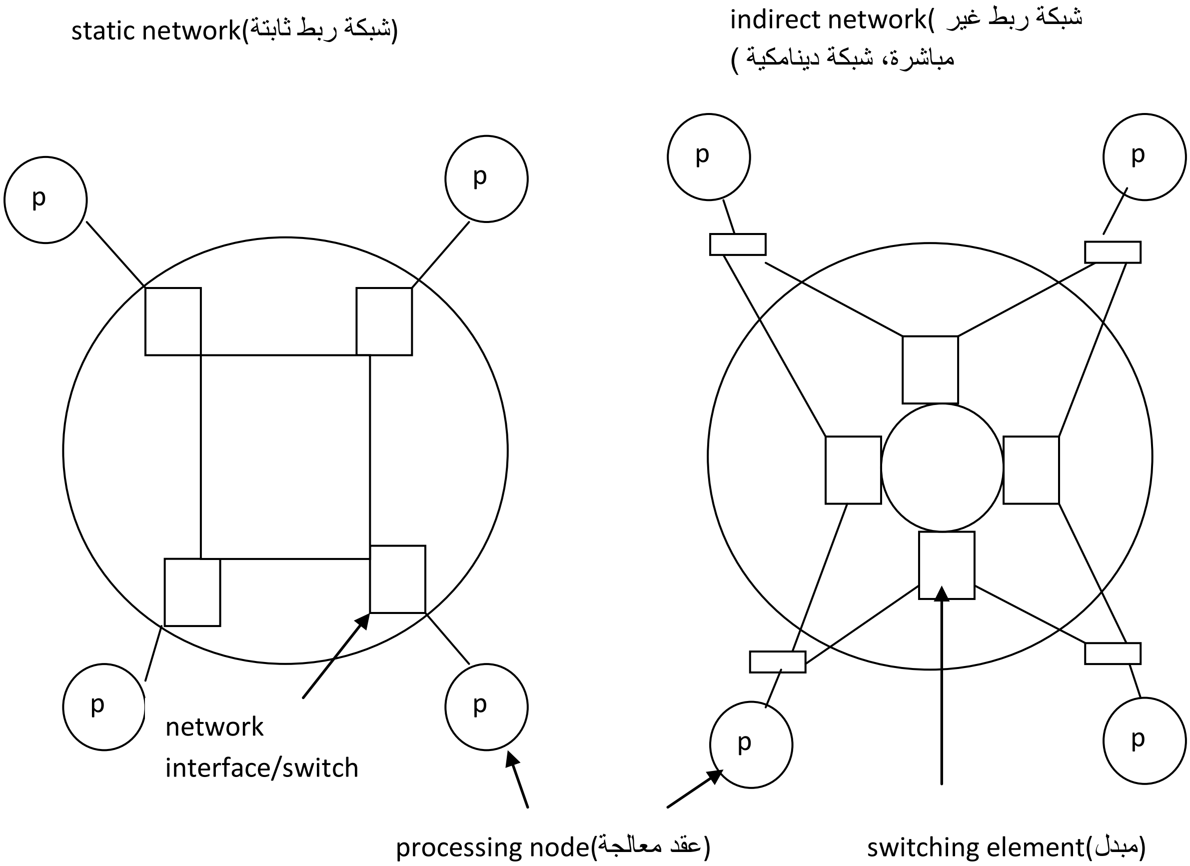
---

[30] :Fayez Gebali, **"Algorithms and Parallel Computing"**, p.83

[31] :Loic Gouarin&Violaine Louvet, " Généralités sur le parallelisme", 2008, p.69

[32]: Vipin Kumar, George Karypus ، مصدر سبق ذكره ، ص59

الشكل رقم (2–12): تصنيف شبكات الربط (من أربع معالجات)



المصدر : Vipin Kumar, George Karypus، مصدر سبق ذكره ، ص60

2. 4 أنواع شبكات الربط: تتميز شبكات الربط فيما بينها من حيث طريقة ربط عقدها أو الطوبولوجيا المستعملة و من أشهرها نذكر:

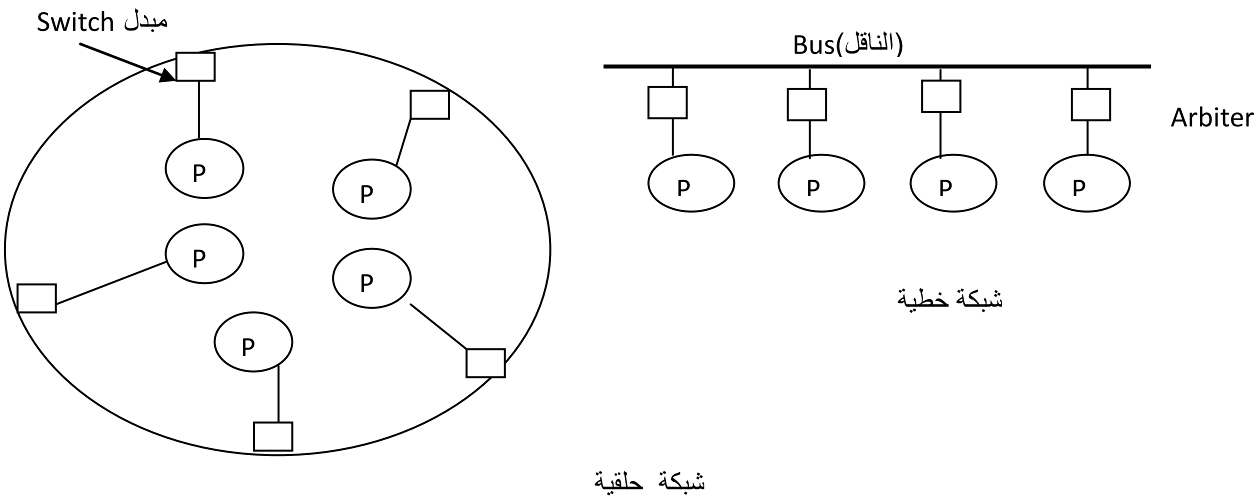
1.4.2 الشبكبة الخطية و الحلقية: تتكون الشبكة الخطية من عقد ، كل عقدة لها عقدتين مجاورتين ، و احدة على اليمين و الأخرى على اليسار. أما العقد الطرفية في الشبكة فلها عقدة واحدة لكل منهما. و بخصوص قطر الشبكة الخطية فيساوي (2p–1) مع أعتبار p هو عدد العقد [33].

وتتميز الشبكة الخطية كذلك خاصة، بعدد المعالجات المتصلة بالناقل فكلما ارتفع عددها نقصت فعاليتها و أصبحت بطيئة و كذلك ببروتكول التحكيم بينها(أولوية ثابتة' أولوية دورية، ولوج عشوائي إلى غير ذلك...) و هذا قصد تفادي الاصطدام أثناء التواصل عبر الناقل. يأخذ التواصل بين كل زوج (Pi, Pj)

[33]: Vipin Kumar, George Karypus ، مصدر سبق ذكره ، ص79

من المعالجات نفس الحيز من الوقت مهما كان موضعها، و يمكن ربط العقد الطرفية للشبكة الخطية و لتكوين شبكة حلقية .[34]

الشكل(2-13): الشبكة الخطية و الحلقية



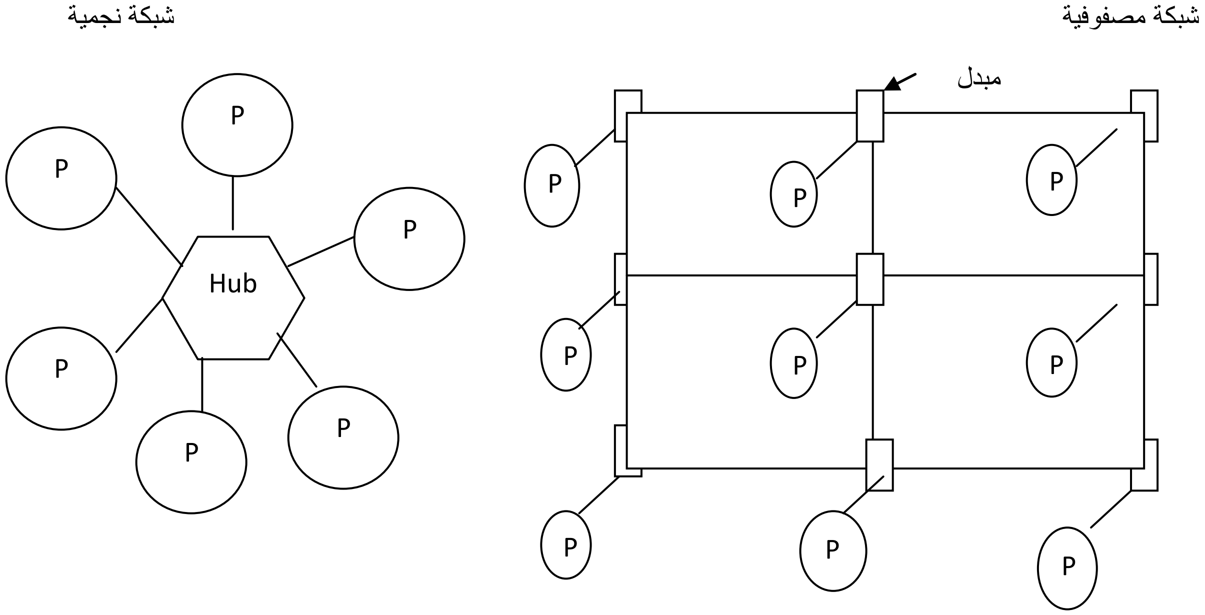
المصدر: **Fayez Gebali, pp. 84-85**

في الشبكة الحلقية، كل معالج (P) متصل بالحلقة عبر المبدل. هذه المبدلات تسمح لأكثر من معالج بارسال و استقبال الرسائل و البيانات في نفس الوقت. يقوم المعالج بارسال بياناته للمبدل المرتبط به و هذا الأخير يرسلها للمبدلات التي في جواره و هكذا حتى تصل هذه البيانات أو الرسالة إلى وجهتها[35].

2.4.2 الشبكة المصفوفية و النجمية:تكون الشبكة المصفوفية على شكل مصفوفة لكل عقدة داخلية أربع عقد مجاورة ، أم العقد الطرفية فدرجتها (2 أو3). أما الشبكة النجمية، فجميع المعالجات فيها مرتبطة بجهاز يسمى المجمع(Hub) يتم بواسطته التواصل و تبادل المعطيات.

[34] :Fayez Gebali, "Algorithms and parallel computing", University Of Victoria, Canada, p84
[35] : Fayez Gebali, "Algorithms and parallel computing", University Of Victoria, Canada, p85

الشكل(2–14): شبكة مصفوفية و نجمية



المصدر: **Fayez Gebali, pp.85-86**

في الشبكة النجمية، جميع المعالجات متصلة بمجمع مركزي (Hub) وكل تواصل بين المعالجات لابد أن يمر عبره. تقل نجاعة هذه الشبكة عندما تتواصل مع جميع المعالجات. في الشبكة المصفوفية يتم التواصل بين المعالجات عبر خوارزمي التوجيه المعد لكل مبدل (switch) أو موجه (routers)[36].

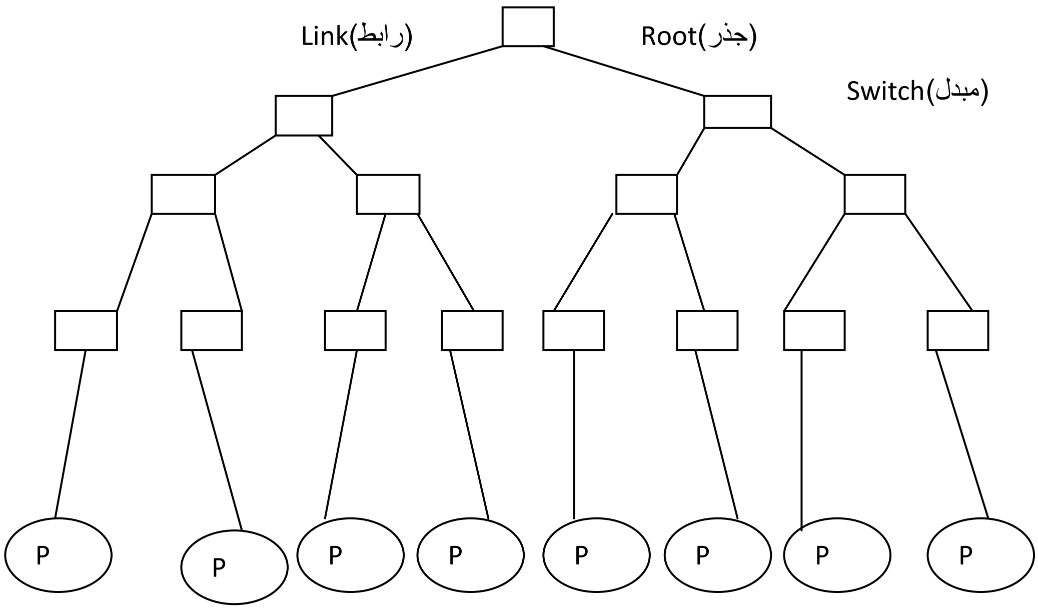
تستخدم هذه الشبكات كثيرا في الحاسبات المتوازية الحالية نظرا لمرونة الاتصال بين العقد.

3.4.2 الشبكات الشجرية

تتلائم هذه التبولوجيا مع البرمجة المتوازية الديناميكية (عدد المهمات غير محدد مسبقا) ومع البرامج من النمط السيد والخدم أو تلك التي تستثمر التوازي على عدة مستويات: مستوى الإجرائية و مستوى التعليمة مثلا و يخصص كل مستوى من الشجرة لمعالجة مستوى من التوازي.

[36]: **Fayez Gebali ، مصدر سبق ذكره، ص85**

الشكل(2-15): شبكة شجرية



Fayez Gebali, p. 90 :المصدر

يظهر في الشكل(2-15) أن الشبكة الشجرية الثنائية المكونة من P معالج تتطلب 2p-1 مبدل أو موجه للتواصل. المعالجات تكون في أوراق الشجرة. المبدلات لها ثلاث روابط ما عدا مبدل جذر الشجرة و مبدل الأوراق و أن قطرها يساوي $2\log_2 p$[37].

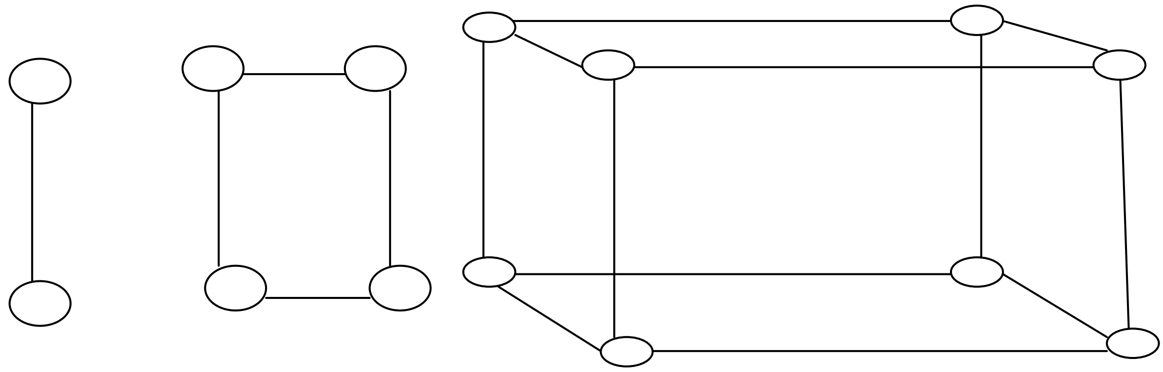
4.4.2 الشبكات المكعبية

تتكون الشبكة المكعبية ذات البعد d من 2^d عقدة، في البعد واحد لدينا عقدتين و في البعد إثنان لدينا 4 عقد و هكذا. تكون لكل عقدة، d عقدة مجاورة ($\log 2^d$)، تكون المسافة بين كل عقدتين على الأكثر logp حيث p هو عدد العقد المكونة للشبكة، كما تساوي درجة الشبكة أو بعدها d=logp و طول قطرها يساوي logp. تحسب المسافة بين عقدتين في الشبكة بعدد وحدات الإستعلام (bits) التي تختلف فيها هاتين العقدتين. و من أهم مميزات هذه التبولوجيا هو قطرها الصغير مقارنة بعدد عقدها [38].

[37] : Fayez Gebali, "Algorithms and parallel computing", University Of Victoria, Canada, p.90

[38]: Vipin Kumar, George Karypis ، مصدر سبق ذكره ، ص83

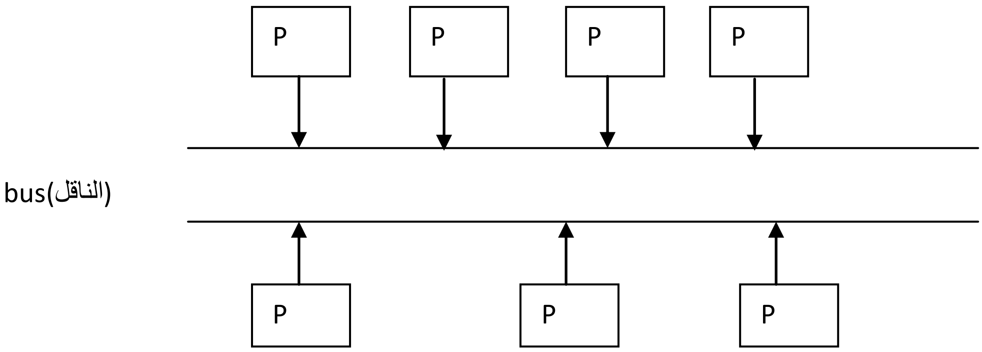
الشكل(2−16): شبكات مكعبية a، b، و c، من واحد ،إثنين و ثلاث أبعاد



المصدر: Vipin Kumar, George Karypis ، مصدر سبق ذكره ، ص82

5.4.2 شبكة الناقل (Bus network): تسمح شبكات الربط الديناميكية في بنيتها إما استعمال الناقل أو المبدل. ويعتبر الناقل عنصر الاتصال المشترك و الوحيد بين المعالجات أو بين المعالجات من جهة و الذاكرة من جهة أخرى في الشبكة. ولتنظيم التواصل المتشرك لا بد من وجود عدد من الخوارزميات التي تمكن من تنفيذ عمليات التراسل المختلفة بين المعالجات المتعددة [39].

الشكل(2−16−أ): شبكة الناقل



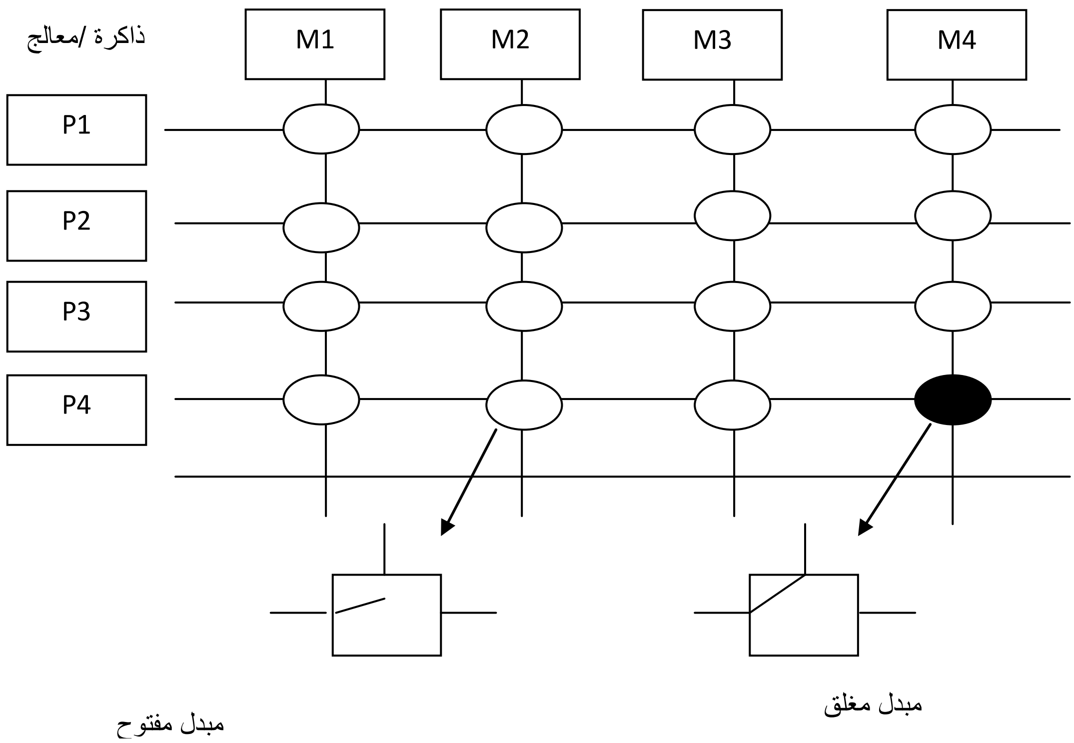
المصدر: « The essentials of computer Organisation and architecture » Linda Null & Julia Lobur
Jones and Bartlett publishers 2003, p. 427

[39]: **Linda Null & Julia Lobur ، مصدر سبق ذكره ، ص.427**

5.4.2 شبكة المبدلات (Crossbar network):

تستخدم شبكة المبدلات مفاتيح التبديل لتغيير توجيه نقل المعلومات بين المعالجات بشكل ديناميكي. أي يحدد مسار نقل المعلومات عند تنفيذ عملية التراسل. و يوجد نوعان من مفاتيح التبديل: مفاتيح crossbar switch الشكل(2–16–ب) ومفاتيح 2x2 switch. تسمح مفاتيح النوع الأول بفتح أو غلق المبدل. كل معالج يمكنه الإتصال بمعالج آخر فقط بغلق مفتاح التبديل بينهما. أما النوع الثاني، فهو شبيه الأول و لكن يمكنه توجيه مدخلاته إلى مختلف الوجهات. في هذا النوع، المبدل له مدخلين و مخرجين و يمكنه أن يكون في إحدى الوضعيات المبينة في الشكل (2–16–ج) [40].

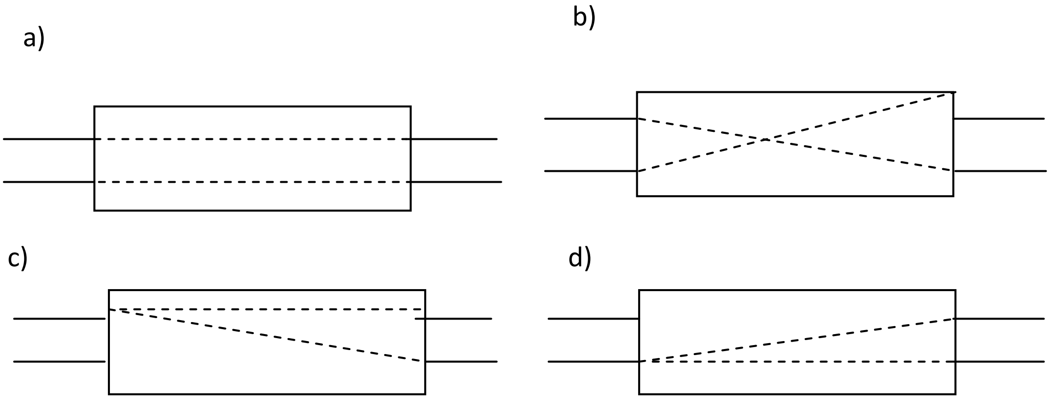
الشكل(2–16–ب): شبكة المبدلات (Crossbar network)



المصدر: Linda Null ، مصدر سبق ذكره، ص427

[40]: Linda Null & Julia Lobur , "The essentials of computer organisation and architecture ", 2003, pp.427–428

الشكل(2−16−ج):وضعيات مفاتيح التبديل (2x2 switch)



a) through(عبرمباشر)　　b) cross(عبر معاكس)

c) upper broadcast(بث علوي)　　d) Lower broadcast(بث سفلي)

المصدر: Linda Null ، مصدر سبق ذكره، ص427

في الوضعية a ، توجه المدخلات من المدخل العلوي إلى المخرج العلوي. و من المدخل السفلي إلى المخرج السفلي.

في الوضعية b، توجه المدخلات من المدخل العلوي إلى المخرج السفلي. و من المدخل السفلي إلى المخرج العلوي.

في الوضعية c، توجه المدخلات من المدخل العلوي إلى كل من المخرج العلوي و المخرج السفلي.

و أخيرا في الوضعية d، توجه المدخلات من المدخل السفلي إلى كل من المخرج العلوي و المخرج السفلي.

2. 5 تقييم شبكات الربط : تقيم شبكات الربط سواء منها الثابتة أو الديناميكية من خلال عدة خصائص أهمها : القطر، عرض المقطع و التكلفة. فالقطر هو أقصى مسافة بين عقدتين (معالجين) في الشبكة. في الشبكة الخطية قطرها هو (p−1)، وفي الشبكة المصفوفية يساوي $2\sqrt{(p-1)}$ ، أما في الشبكة الشجرية و المكعبية فهو log (p) ، حيث p هو عدد العقد في الشبكة. أما عرض المقطع فهو أدنى عدد الأسلاك في الشبكة عندما يتم تقسيمها إلى قسمين. ففي الشبكة الخطية، عرض المقطع هو واحد، و في

الشبكة المصفوفية هو $\sqrt{p}$، أما في الشبكة المكعبية فيساوي $(p^2/4)$. و فيما يخص التكلفة المتمثلة في عدد خطوط الربط أو عدد خطوط المبدلات (حسب الحالة) ففي الشبكة الخطية هو $(p-1)$ ، و في الشيكة المصفوفية فهو $2(p-\sqrt{p})$ و أما في الشبكة المكعبية فيساوي $(plogp)/2$. بالإضافة إلى هذا تدخل عوامل أخرى في تقييم نجاعة شبكات الربط كطول و سعة الأسلاك و غير ذلك[41].

2. 6 نماذج الخوارزميات المتوازية:

يتم تطوير نموذج خوارزمية متوازية من خلال النظر في استراتجية تقسيم البيانات و طريقة المعالجة و تطبيق استراتجية مناسبة للحد من التفاعلات. و في ما يلي نذكر النماذج التالية[42]:

و لمزيد من التوضيح أنظر كذلك المرجع.[43]

- نموذج توازي البيانات أو المعطيات(data parallel model)
- نموذج الرسم البياني للمهام(task graph model)
- نموذج تجميع العمل(work pool model)
- نموذج السيد و العبد(master–slave model)
- نموذج المنتج و المستهلك أو نموذج خط الأنابيب(producer–consumer model)
- نموذج الهجين (hybrid models)

أ– **نموذج توازي المعطيات:**في نموذج توازي المعطيات، يتم تعيين المهام إلى المعالجات بطريقة ستاتكية (أي المعالجات محددة مسبقا) و كل مهمة تنفذ أنواع مماثلة من عمليات على معطيات مختلفة. توازي المعطيات هي نتيجة لعملية واحدة يتم تطبيقها على معطيات متعددة. و يمكن تطبيق نموذج توازي المعطيات على فضاءات العناوين المشتركة (ذاكرة مشتركة) و نماذج تمرير الرسائل. و أهم ميزة لنموذج توازي البيانات هي أن كثافة التوازي تزداد مع حجم معطيات المسألة . يتناسب هذا النوع من النماذج

---

41 : **Vipin Kumar,"Introduction to parallel computing", p.87-89**

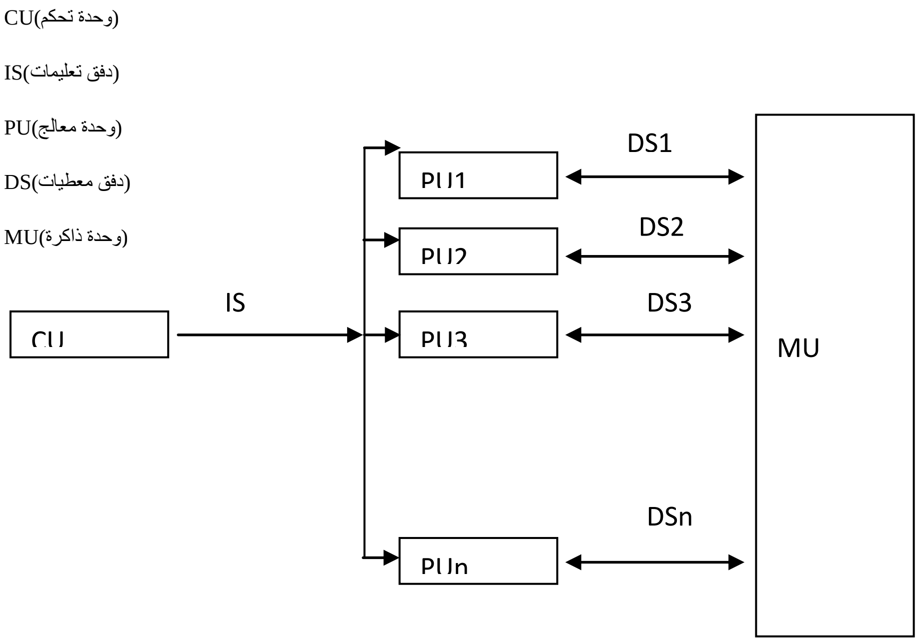
42 : **Parallel Algorithm–Quick Guide, pp5–10**

http://www.tutorialspoint.com/parallel_algorithm_models.htm,

43 :Vipin Kumar,"Introduction to parallel computing", pp.85-86

على الحاسبات المتوازية SIMD . و كمثال عن ذلك خوارزمية ضرب المصفوفات الكثيفة و كبيرة الحجم.

الشكل (2−17) : نموذج توازي المعطيات

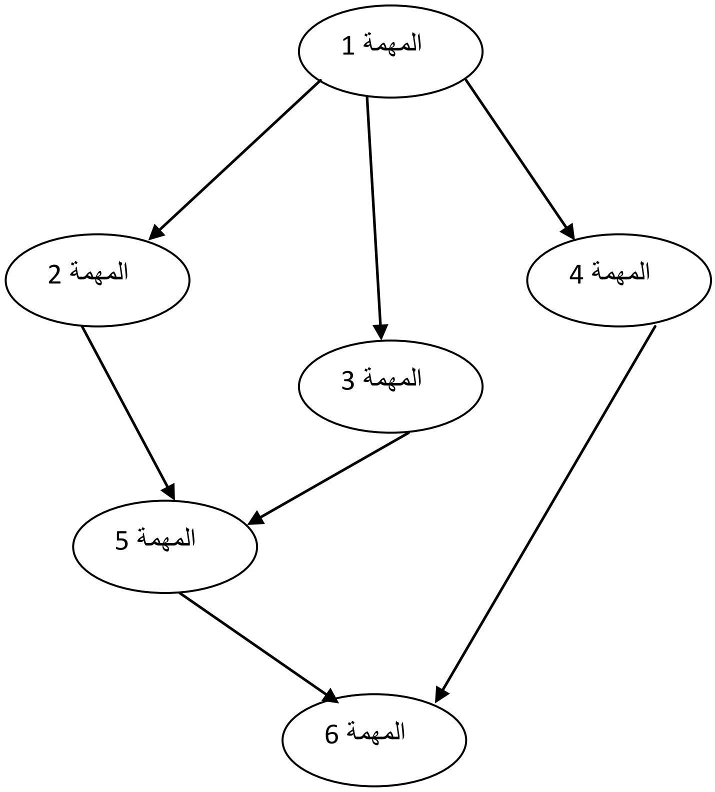


المصدر : http://www.tutorialspoint.com/parallel_algorithm_models.htm ، **مصدر سبق ذكره ، ص6**

ب− **نموذج الرسم البياني للمهام :**في نموذج الرسم البياني للمهام، يتم التعبير عن التوازي بواسطة رسم بياني للمهام. يتم استخدام الترابط بين المهام لتقليل تكاليف التفاعل. كما يتم فرض هذا النموذج عندما يكون فيها كمية البيانات المرتبطة بالمهام ضخمة بالمقارنة مع عدد الحسابات المرتبطة معهم. و يتم تعين المهام للمساعدة في تحسين كلفة حركة البيانات بين المهام.

مثال: الترتيب السريع المتوازي، و الخوارزميات الناتجة من طريقة فرق تسد هنا، تقسم المسائل إلى مهام صغيرة جدا و تنفيذها كرسم بياني. كل مهمة تعتبر وحدة مستقلة من الوظائف التي لها تبعيات على واحدة أو أكثر من مهمة سابقة. بعد الانتهاء من مهمة، يتم تمرير مخرج المهمة السابقة إلى المهمة التابعة. لا تبدأ مهمة في التنفيذ إلا بعد أن تنتهي سابقتها من ذلك.

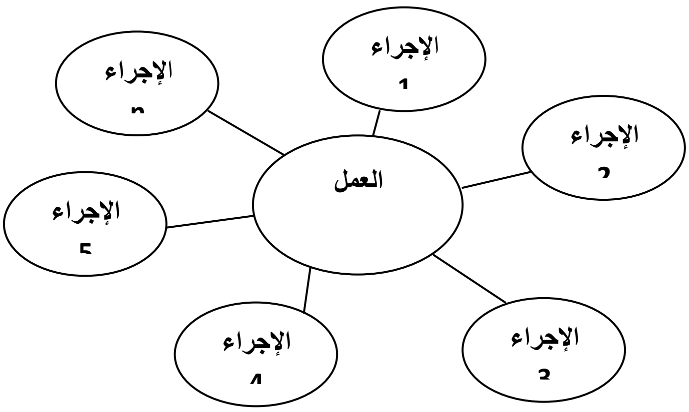
الشكل (2–18) : نموذج الرسم البياني للمهام



المصدر: http://www.tutorialspoint.com/parallel_algorithm_models.htm ، مصدر سبق ذكره ، ص6

ج- نموذج تجميع العمل :في نموذج تجميع العمل، يتم تعيين المهام ديناميكيا. و يستخدم هذا النموذج عندما تكون كمية من البيانات المرتبطة بالمهام هي أصغر نسبيا من الحسابات المرتبطة معها. لا يوجد تعيين مسبق للمعالجات على المهام وقد يكون مركزيا أو لامركزيا.مثال: البحث الشجيري المتوازي.

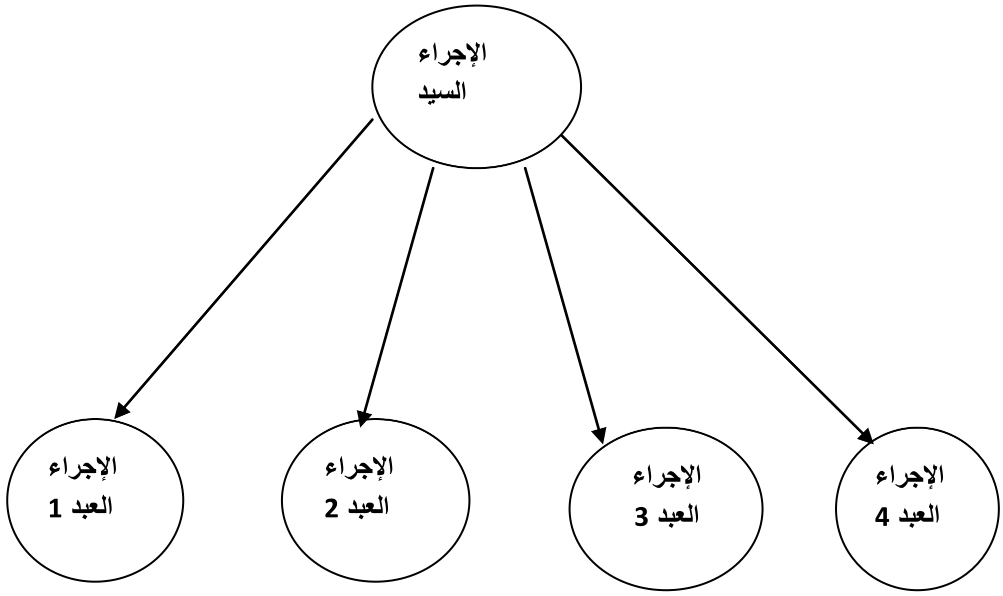
الشكل (2–19) : نموذج تجميع العمل



المصدر: http://www.tutorialspoint.com/parallel_algorithm_models.htm ، مصدر سبق ذكره ، ص7

د- نموذج السيد و العبد : في النموذج السيد و العبد، واحد أو أكثر من الإجرائيات الرئيسية تولد مهمة و توكلها إلى الإجراء العبد. يتم هذا التوكيل ستاتكيا أو دينامكيا. يمكن للسيد تقدير حجم المهام، أو يمكن تعيين عشوائي لموازنة الأعباء. هذا النموذج يستعمل سواء في حالة فضاء تقسيم العناوين أو في حالة تمرير الرسائل لأن التفاعل بين المهام يتم في الطريقتين.

الشكل (2-20): نموذج السيد و العبد

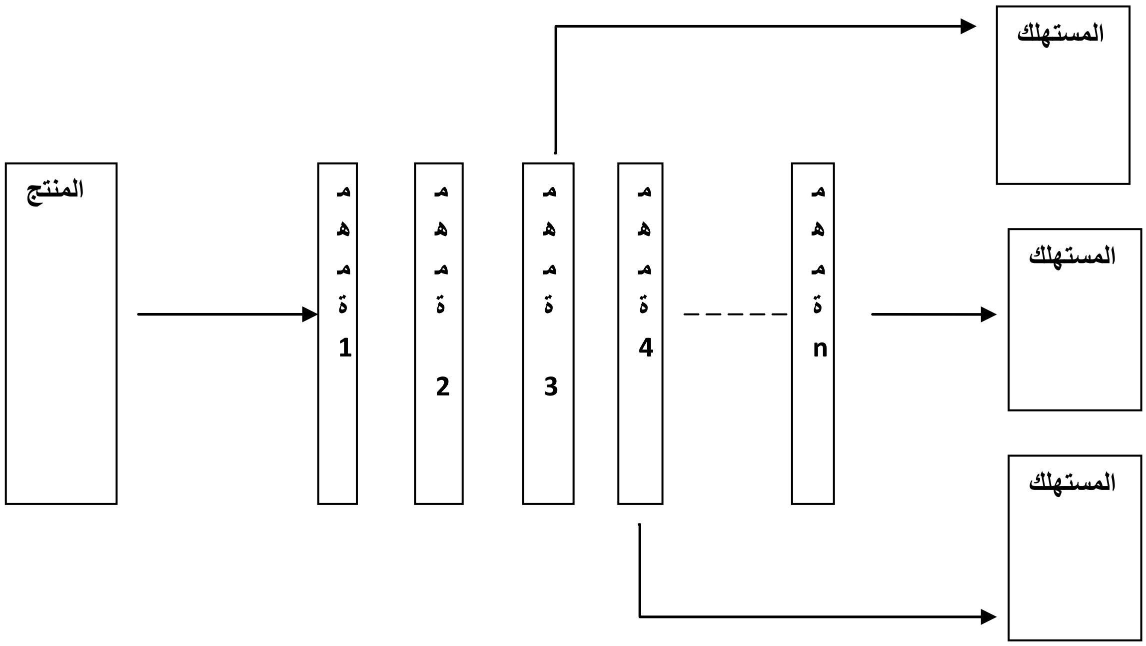


المصدر: http://www.tutorialspoint.com/parallel_algorithm_models.htm **، مصدر سبق ذكره ، ص8**

ه- نموذج المنتج و المستهلك: يعرف كذلك بنموذج خط الأنابيب. في هذا النموذج يتم تمرير مجموعة من البيانات عبر سلسلة من الإجراءات، كل منها يؤدي مهمة معينة. هنا وصول بيانات جديدة يولد تنفيذ مهمة جديدة بواسطة إجراء في قائمة الإنتظار. يمكن أن تشكل هذه الإجراءات قائمة انتظار على شكل جداول خطية أو متعددة الأبعاد أو رسوم بيانية من أو دون دارات.

هذا النموذج هو سلسلة من المنتجين و المستهلكين و كل إجراء في قائمة الإنتظار يعتبر مستهلك لمقطع من المعطيات للإجراء الذي سبقه في قائمة الإنتظار وكمنتج للمعطيات للإجراء الذي يليه في القائمة.

الشكل (2-21): نموذج المنتج و المستهلك



المصدر: http://www.tutorialspoint.com/parallel_algorithm_models.htm ، مصدر سبق ذكره ، ص9

و- **النموذج الهجين** :عندما يكون هناك حاجة إلى أكثر من نموذج واحد لحل مسألة معينة. و قد يتكون النموذج المختلط من نماذج متعددة على شكل هرمي أو متعدد النماذج تطبق بالتتابع لمختلف مراحل الخوارزمي المتوازي، مثل خوارزمي الفرز السريع المتوازي.

خلاصة :

تعرضنا في الفصل الثاني إلى تصنيف الحاسبات. إذ قسم فلاين الحاسبات إلى أربع فيئات و هذا حسب توازي المعطيات أو توازي التعليمات وهي:

– الحاسبات وحيدة تدفق التعليمات و وحيدة تدفق المعطيات(SISD)، وتتم المعالجة بها بالتسلسل أي تنفذ تعليمة واحدة على معلمة واحدة في كل لحظة. هذه الحاسبات هي تلك الحاسبات التقليدية المعروفة للجميع.

– الحاسبات وحيدة تدفق التعليمات ومتعددة تدفق المعطيات (SIMD). تنفذ التعليمة الواحدة على معطيات مختلفة في كل لحظة. كما يمكن كذلك تقسيم هذه الحاسبات (SIMD) إلى أربع وهذا حسب طريقة الولوج للذاكرة و هي:

- ✓ حاسبات SM SIMD (EREW) محدودة القراءة ومحدودة الكتابة.
- ✓ حاسبات SM SIMD (CREW) متعددة القراءة و محدودة الكتابة.
- ✓ حاسبات SM SIMD (ERCW) محدودة القراءة و متعددة الكتابة.
- ✓ حاسبات SM SIMD (CREC) متعددة القراءة و متعددة الكتابة.

– حاسبات متعددة تدفق التعليمات و متعددة تدفق المعطيات (MIMD). يتم في هذا النوع من الحاسبات تنفيذ عدة تعليمات على معطيات مختلفة خلال الدورة الزمنية للحاسب. و يوجد بهذه الحاسبات كذلك فيأتين فرعيتين حسب طريقة التواصل بين المعالجات، فنجد حاسبات (MIMD) ذاكرة مشتركة (SM MIMD) و حاسبات (MIMD) شبكة اتصالات (Interconnection Network MIMD).

ثم تطرقنا بعد ذلك إلى مختلف أنواع هذه الشبكات من حيث الطوبولوجيا نذكر مثلا:الشبكات الخطية، الشبكات الحلقية، الشبكات المصفوفية، الشبكات الشجرية و الشبكات المكعبية. بعد ذلك قمنا بتعريف مختلف نماذج الخوارزميات المتوازية أو كما يسمى أيضا بنماذج التنفيذ المتوازي مثل نموذج توازي المعطيات، نموذج الرسم البياني للمهام، نموذج السيد و العبد و أخيرا النموذج الهجين. ووضحنا كل هذه النماذج بأشكال.

الفصل الثالث

تصميم الخوارزميات المتوازية

تمهيد : لقد تطور علم الخوارزميات في حل المسائل و هذا راجع لتطور الحاسبات إذ أصبحت في وقتنا الراهن تحتوي على العديد من المعالجات و كل معالج بدوره يحتوي على العديد من النوى مما يمكن من المعالجة المتعددة . و نميز ثلاثة أنواع من الخوارزميات، خوارزميات تسلسلية ، خوارزميات متوازية و خوارزميات مختلطة و هذه الأخيرة تحتوي على أجزاء مستقلة قد تنفذ بالتوازي . فالخوارزمي المتسلسل هو الذي تنفذ مهامه بالتتابع الواحدة تلوى الأخرى و هذا بسبب ارتباط المعطيات بها، في حين الخوارزمي المتوازي متكون من مهام مستقلة عن بعضها و يمكن تنفذها في نفس الوقت لكون المعطيات بها غير مرتبطة ، أما الخوارزمي المختلط (متسلسل/متوازي) فهو ذلك الخوارزمي الذي تجمع مهامه في مراحل مختلفة بحيث يمكن تنفيذ مهام كل مرحلة بالتوازي و لكن تنفذ المراحل بالتسلسل. ينفذ الخوارزمي المتسلسل على حاسب تسلسلي مكون من معالج واحد ، بينما ينفذ الخوارزمي المتوازي على حاسوب متعدد المعالجات [1].

يعتبر تصميم الخوارزميات المتوازية أكثر تعقيدا من الخوارزميات المتسلسلة لأنه يتطلب الأخذ بعين الاعتبار عدة عوامل كجزء الخوارزمي الذي يمكن معالجته بالتوازي، كيفية توزيع المعطيات، تتابع أو ارتباط المعطيات وكذلك التزامن بين المعالجات وغير ذلك.

1.3 طرق تصميم الخوارزميات المتوازية:

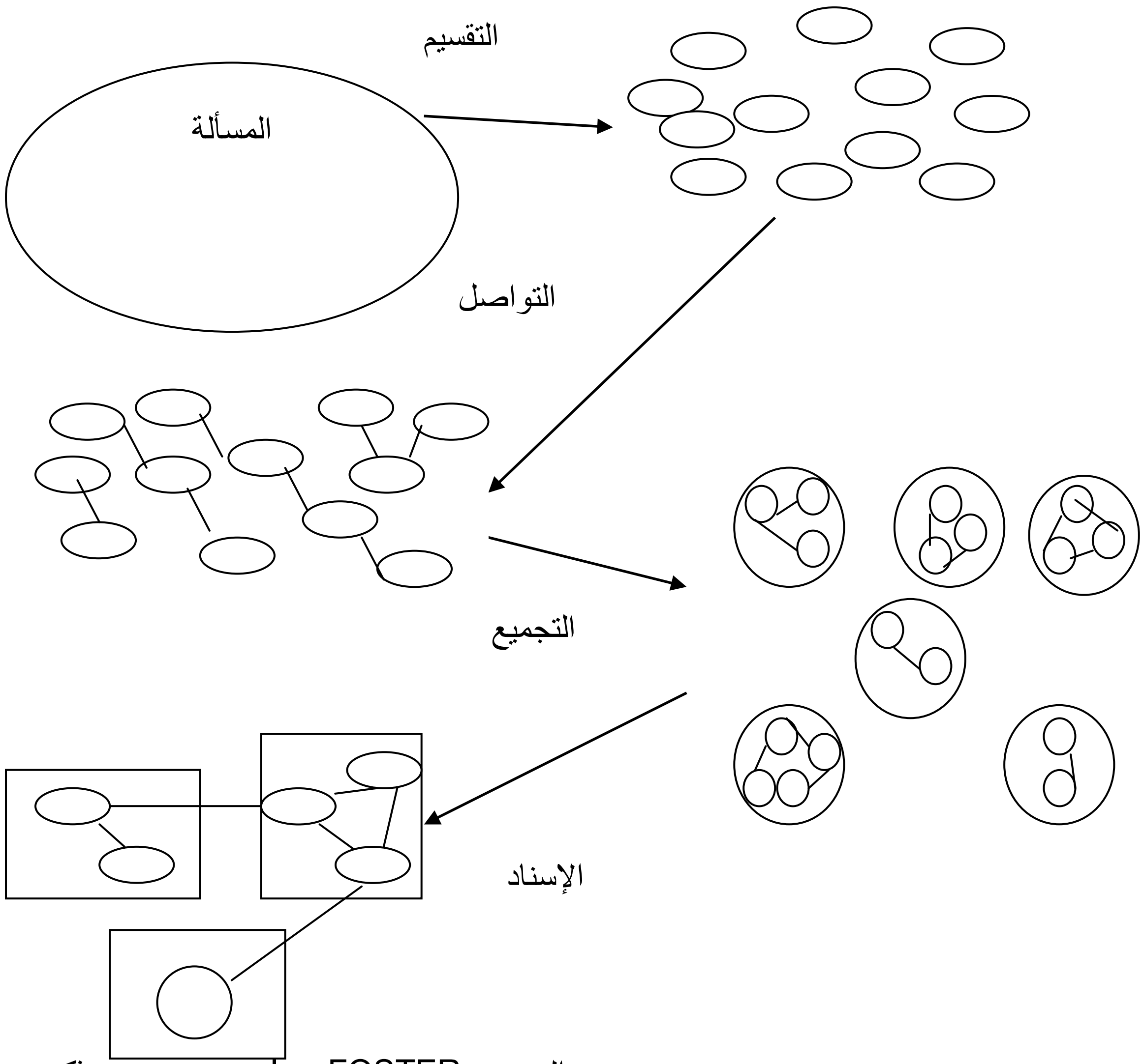
يستخدم في تصميم الخوارزميات المتوازية طريقتان أساسيتان هما :

الأولى تتمثل في تحديد واستغلال التوازي الموجود ضمنيا في الخوارزمي المتسلسل ، الثانية تتمثل في ابتكار خوارزمي متوازي جديد.[2] ففي الطريقة الأولى تقسم المسألة إلى مسائل جزئية صغيرة تسند إلى عدة معالجات قصد تنفيذها ثم بعد ذلك استخلاص الحل النهائي للمسألة. يدخل في هذا التقسيم نوعان و هما تقسيم البيانات و تقسيم المهام. التقسيم مرتبط خاصة بتنظيم الذاكرة (ذاكرة موزعة، ذاكرة مشتركة أو ذاكرة مختلطة بينهما). و يمكن إضافة شرط إضافي آخر و هو نجاعة التواصل بين المعالجات و التي تتطلب من المصمم اختيار شكل شبكة الربط التي يتم من خلالها تزامن هذا التواصل و طريقة تبادل البيانات بينها [3].

[1] : Fayez Gebali, "Algorithms and Parallel Computing", University of Victoria, Canada, p.8
[2] :Rédha LOUCIF, "Parallélisation d'Algorithmes d'Optimisation Combinatoire", 2014, p.31
[3]:Fayez Gebali ، مصدر سبق ذكره ، ص12

في هذا الغرض عرض الباحث Ian FOSTER طريقة من أربع مراحل لتصميم الخوارزميات المتوازية[4].

الشكل (3) : طريقة FOSTER لتصميم الخوارزميات المتوازية:



المصدر: Ian FOSTER ، مصدر سبق ذكره، ص28

المرحلة الأولى هي تقسيم المسألة و معطياتها إلى أجزاء صغيرة تسمى مهام يمكن حلها بالتوازي. و يمكن لحجم هذه المهام أن يكون متساوي أو مختلف. و يمكن توضيح التقسيم من خلال بيان ، عقده تمثل المهام و أضلاعه التتابع بينها.

المرحلة الثانية تتمثل في تحديد التواصل بينها قصد تنسيق تنفيذ المهمة.

[4] : Ian FOSTER, "Introduction to parallel computing", p27

المرحلة الثالثة يتم تقييم عمليات الحساب و التواصل المعرفتين في المرحلتين السابقتين بالنسبة لمتطلبات النجاعة و تكاليف تشغيلها. كما يمكن تجميع المهام الصغيرة في مهام أكثر أهمية إن اقتضت الضرورة لذلك.

المرحلة الرابعة يتم اسناد كل مهمة لإجراء قصد تحقيق الأهداف التنافسية و المتمثلة في تعظيم استعمال المعالج و تدنئة تكاليف التواصل. يمكن أن يكون الاسناد ستاتكيا أو يتم تحديده ديناماكيا أي أثناء التنفيذ من خلال خوارزميات مخصصة لهذا الغرض[5].

و سوف نعرض لاحقا أمثلة لتوضيح طريقة تصميم الخوارزميات المتوازية. المثال الأول هو ضرب مصفوفة بشعاع ، والمثال الثاني يتمثل في عملية الاستعلام في قواعد البيانات ، بالإضافة إلى تعريف عدد من المفاهيم المستخدمة في طريقة التصميم.

3-2 مفاهيم التصميم :

- **بيان التبعية(الترابط) (Dependency Graph):** هو بيان غير موجه ، يستخدم للتعبير عن التبعية فيما بين المهام و الترتيب النسبي للتنفيذ. تمثل عقد البيان المهام و الأضلاع تمثل المعطيات المستعملة من طرف المهام. هذه المعطيات يمكن أن تكون نتائج إدخال ، إخراج أو داخلية . وبما أن هذه الأضلاع ليست موجهة و بالتالي تشير إلى أي تبعية لمعطيات إدخال أو إخراج.
- **بيان السوابق (precedence graph):** هو بيان موجه بدون دارة. عقد البيان تمثل مهام البرنامج و الأضلاع تمثل تبعية المعطيات بين المهام. بداية الضلع يمثل مخرج مهمة و نهاية الضلع مدخل لمهمة. في مثل هذا البيان يبدأ تنفيذ مهمة عندما تنتهي من التنفيذ جميع المهمات التي سبقتها[6].
- **بيان تدفق المعطيات:** تعتبر الارتباطات بين المهام تبادل للمعطيات، ومن الممكن إضافة معلومات للمعطيات المتبادلة في البيان:
  - القراءة فقط: المهمة لا يمكنها تغيير المعلم.
  - الكتابة فقط: المهمة يمكنها تعيين المعلم.

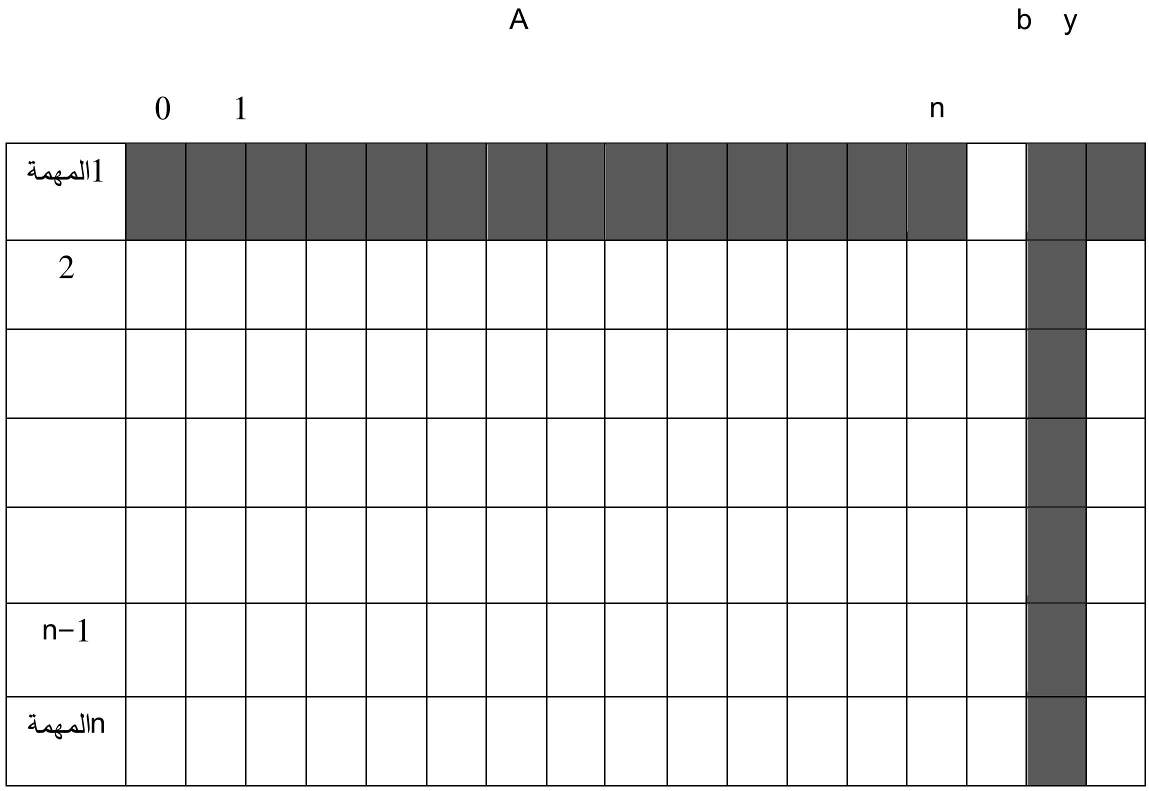
[5]: Ian FOSTER ، مصدر سبق ذكره ، ص29

[6]: Fayez Gebali ، مصدر سبق ذكره ، صص.4-5

– التغيير: يمكن للمهمة تغيير المعلم.

مثال (3−1): ضرب مصفوفة بشعاع:في هذا المثال نريد إجراء عملية ضرب مصفوفة A بحجم nxn مع الشعاع b ، فنحصل على شعاع آخر y. يمثل y(i) ناتجً ضرب السطر i من A مع كامل الشعاع b. و الشكل (3−1) يوضح عملية حساب كل قيمة y(i) و التي يمكن اعتبارها مهمة.

الشكل (3−1): جداء مصفوفة A بشعاع b مقسمة إلى n مهمة، حيث n هي عدد أسطر المصفوفة. الجزء الذي تتعامل معه (مدخلات و مخرجات) المهمة 1 موضح باللون الغامق.



المصدر1: **Introduction to parallel Algorithms and Parallel Program design**", University of Oregon , p. 17

المصدر2: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, "**Introduction to parallel Computing:Principles of parallel Algorithm Design**", Addison Wesley, 2003, p.5

نلاحظ بأن جميع المهام المحددة في الشكل (3−1) هي مهام مستقلة و يمكن تنفيذها سوياً أو على أي تسلسل. فكل عنصر y(i) من الشعاع y يمكن حسابه بصفة مستقلة عن العناصر الأخرى و بالتالي اعتبار حسابه مهمة مستقلة . و بشكل عام ، في بعض المسائل قد تكون بعض المهام فيها بحاجة إلى بيانات ناتجة عن مهام أخرى و لذا فإن عليها الانتظار إلى أن تُنهي هذه المهام أعمالها.

و في بيان التبعية تعتبر **العقد** كمهام ، أما الخطوط التي تصل بين العقد (يطلق عليها **أضلاع**) فتدل على التتابع بين المهام. فالمهمة التي تتطابق مع أحد العقد لا يمكن تنفيذها إلا حين انتهاء تنفيذ جميع المهام التي تدخل إليها (أي أن الخط الواصل يكون داخل للمهمة و ليس خارجا منها).

مثال (3−2) إجرائية الاستعلام من قواعد البيانات:يوجد في الجدول (3−1) عرض لقاعدة بيانات خاصة بسيارات ، و كل صف في هذا الجدول هو سجل يحتوي على بيانات سيارة محددة ، مثل المعَرِّف ID ، و سنة الإنتاج year ، و اللون color ، الخ ...

الجدول (3−1): قاعدة بيانات لتخزين معلومات عن السيارات.

ID(المعرف)	Model(النوع)	Year(سنة السير)	Color(اللون)	Price(السعر): $
4523	Civic	2003	Blue	55,000
3476	Corolla	1999	White	45,000
7623	Camry	2003	Green	59,500
9834	Prius	2001	Green	48,000
6734	Civic	2001	White	47,000
5342	Altima	2001	Green	49,000
3845	Maxima	2001	Blue	52,000
8354	Accord	2000	Green	48,000
4395	Civic	2001	Red	47,000
7352	Civic	2002	Red	48,000

المصدر: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, "**Introduction to parallel Computing:Principles of parallel Algorithm Design**", Addison Wesley, 2003, p.7

لنفترض أننا نريد تنفيذ الاستعلام التالي:

MODEL= "Civic" AND YEAR= "2001" AND (COLOR="Green" OR COLOR=" White")

يقوم هذا الاستعلام بالبحث عن جميع السيارات التي من نوع Civic و التي أنتجت في السنة 2001 و لها أحد اللونين: الأخضر أو الأبيض.

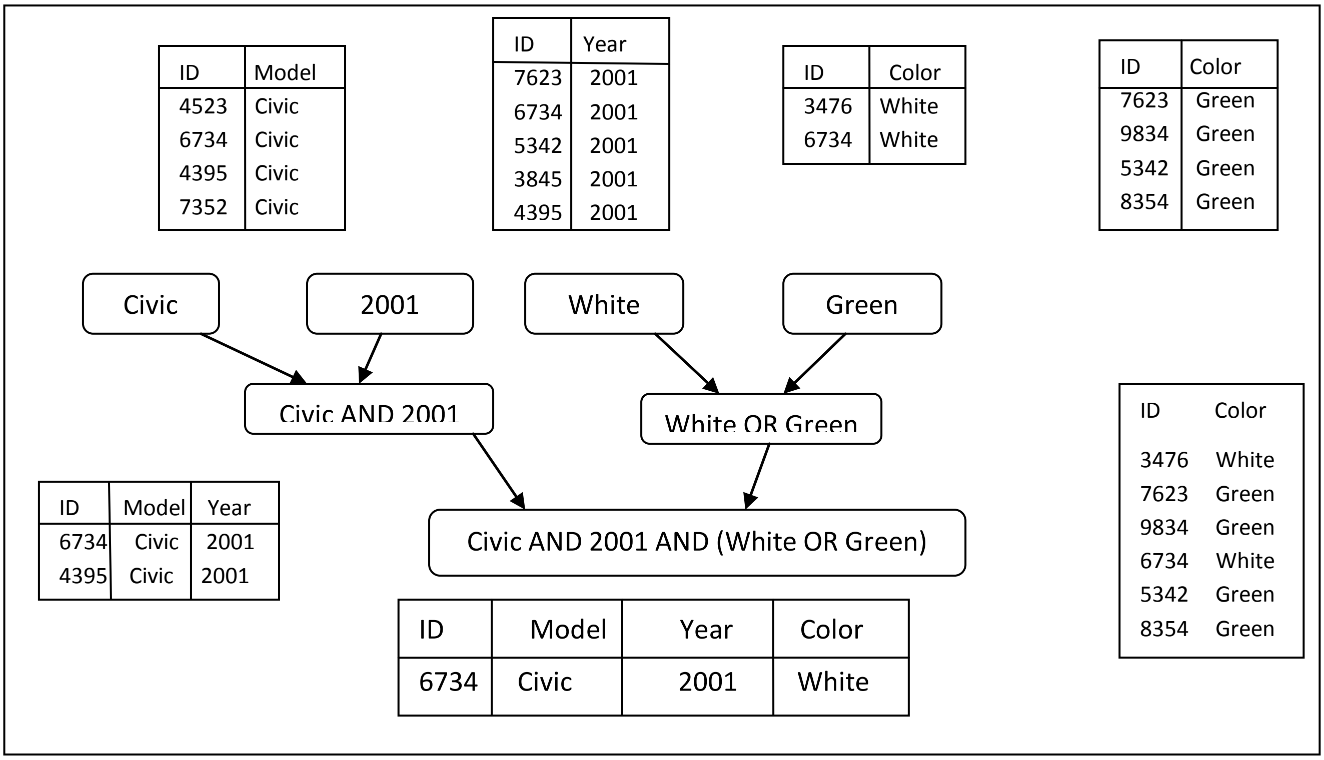
في قواعد البيانات العلائقية (Relational Database)، يتم تنفيذ هذا الاستعلام في ثلاث مراحل. ففي المرحلة الأولى يتم إنشاء الجداول الأربعة التالية:

- ✓ جدول يحتوي على جميع السيارات من نوع Civic.
- ✓ جدول يحتوي على جميع السيارات التي أنتجت في عام 2001.
- ✓ جدول يحتوي على جميع السيارات ذات اللون الأخضر.
- ✓ جدول يحتوي على جميع السيارات ذات اللون الأبيض.

ثم بعد ذلك في المرحة الثاني تتم العملية بواسطة دمج هذه الجداول عن طريق حساب التقاطعات أو الاتحادات بين الجداول مثنى مثنى . و على وجه التحديد سيتم حساب التقاطع للجدولين "السيارات من نوع Civic" و "السيارات التي أنتجت عام 2001" و ذلك لإنشاء جدول يحتوي على سياراتCivic التي أنتجت عام 2001. و بنفس الأسلوب ، سيتم إجراء اتحاد لجدولي "اللون الأخضر" و "اللون الأبيض" و ذلك لكي يتم إنشاء جدول لجميع السيارات ذات اللون الأخضر أو الأبيض. و في المرحلة الثالثة يتم إجراء التقاطع للجدول الذي يحتوي على سيارات 2001 Civic مع الجدول الذي يحتوي على جميع السيارات الخضراء أو البيضاء اللون ، و بذلك يتم الحصول على نتيجة الاستعلام.
يمكن للحسابات المختلفة التي استخدمت لمعالجة الاستعلام في المثال السابق أن تمثل بواسطة مخطط التبعية الموضح في الشكل (2-3)[7].

[7]: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, "**Introduction to parallel Computing:Principles of parallel Algorithm Design**", Addison Wesley, 2003, p.7

الشكل (3−2): الجداول المختلفة ومخطط التبعية في عملية الاستعلام 1.



المصدر: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama ، المصدر سبق ذكره، ص8

الأضلاع الموجهة بين العقد توضح العلاقة (أو التبعية) بين المهام. فعلى سبيل المثال ، قبل حساب الجدول الذي يحتوي على Civic 2001 يجب أولا حساب الجدولين "سيارات Civic" و "سيارات 2001". يوجد عدة طرق للحصول على بعض الحسابات ، و خصوصا تلك التي تستخدم المعاملات المنطقية . مثل: المعامل AND و OR و غيرهما . فالطرق المختلفة لتنفيذ الاستعلام تؤدي إلى مخططات تبعية مختلفة. فعلى سبيل المثال استعلام قاعدة البيانات الواردة في المثال (3−2) يمكن أن يتم حله بالأسلوب الآتي:

أولا: تحديد جدول يحتوي على السيارات ذات اللون الأخضر أو الأبيض.

ثانيا: إجراء تقاطع لجدول "السيارات ذات اللون الأخضر أو الأبيض" مع الجدول: "سيارات أنتجت في عام 2001".

ثالثا: تدمج النتائج مع جدول "سيارات Civic".

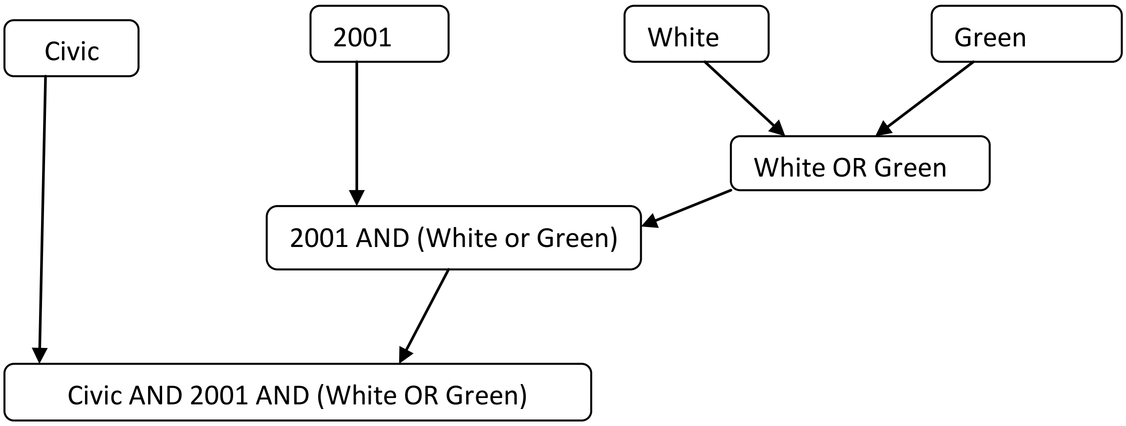
الشكل (3–3): الجداول و مخطط التبعية لعملية الاستعلام 2.

ID	Model
4523	Civic
6734	Civic
4395	Civic
7352	Civic

ID	Year
7623	2001
6734	2001
5342	2001
3845	2001
4395	2001

ID	Color
3476	White
6734	White

ID	Color
7623	Green
9834	Green
5342	Green
8354	Green



ID	Color
3476	White
7623	Green
9834	Green
6734	White
5342	Green
8354	Green

ID	Color	Year
6723	Green	2001
6734	White	2001
5342	Green	2001

ID	Model	Year	Color
6734	Civic	2001	White

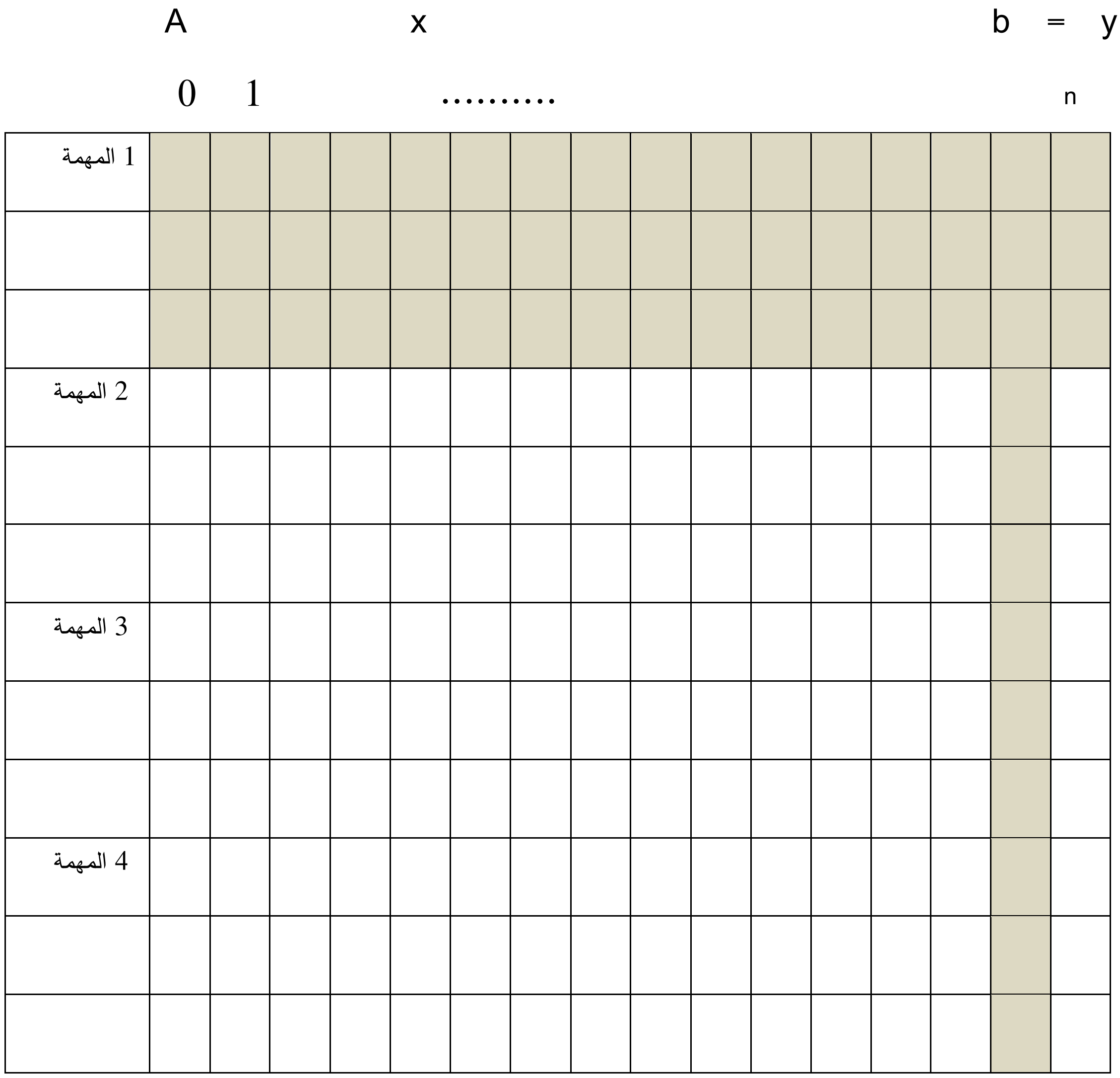
المصدر: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama ، المصدر سبق ذكره ، ص9.

الحبوبية (Granularity): في مجال الخوارزميات يسمى حجم و عدد المهام في المسألة المجزأة إلى عدد كبير من المهام الصغيرة بمصطلح الحبوبية الناعمة ، في حين تقسيم مهام إلى عدد صغير يطلق علية تسمية الحبوبية الخشنة[8].

فعلى سبيل المثال، التقسيم لمسألة عملية ضرب مصفوفة بشعاع الواردة في المثال (3–1) تعتبر تقسيماً حبوبياً ناعماً و ذلك لأن كل مهمة من المهام الكثيرة تقوم بتنفيذ عملية الضرب لسطر واحد. أما في الشكل (3–4) ففيه عرض للتقسيم من نوع الحبوبية الخشنة لنفس المسألة إلى 4 مهام ، بحيث تقوم كل مهمة بتنفيذ n/4 من العمل لكامل الشعاع الناتج. يلجأ للحبوبية الخشنة ، للأن في الحقيقة عدد المعالجات محدود و لا يوافق العدد الكبير للمهام في كل الأحوال و كذلك للتقليل من كلفة التواصل.

[8] Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama: ، مصدر سبق ذكره ، ص9

الشكل (3-4): جداء مصفوفة بشعاع مقسمة إلى 4مهام. الجزء الذي تتعامل معه (مدخلات و مخرجات) للمهمة 1 موضح باللون الغامق. فالمهمة في هذا الشكل تقوم بحساب ثلاث عناصر من الشعاع y والموافقة لضرب ثلاث صفوف من المصفوفة A بالشعاع b.



المصدر1: Introduction to parallel Algorithms and Parallel Program design", University of Oregon , p.18

المصدر2: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, المصدر سبق ذكره، ص.10

إن عدد المهام التي يمكن تنفيذها بالتوازي في أي تقسيم يسمى درجة التزامن. و يمكن لهذا العدد من المهام أن يتغير أثناء تنفيذ البرنامج ، فالدرجة القصوى للتزامن هو العدد الأقصى لهذه المهام في كل لحظة و أثناء التنفيذ. فالدرجة المتوسطة للتزامن هو متوسط عدد المهام التي يمكن معالجتها بالتوازي أثناء تنفيذ البرنامج. ترتفع درجة التزامن كلما أصبح التقسيم أكثر نعومة و العكس. و يعتبر المسار الحرج في بيان تتابع المهام و هو أطول مسار حيث يحدد فيه أقصر وقت ممكن لتنفيذ برنامج بالتوازي [9]. فعلى سبيل المثال ، الدرجة القصوى للتزامن في مخطط التبعية الموضح في الشكل (3-2) و شكل (3-3) هو 4 . و في مخططات التبعية هذه ، تحصل الدرجة القصوى للتزامن في البداية عند ما يتم حساب الجداول الأربعة (النوع ، السنة ، اللون الأبيض ، اللون الخضر) بنفس الوقت.

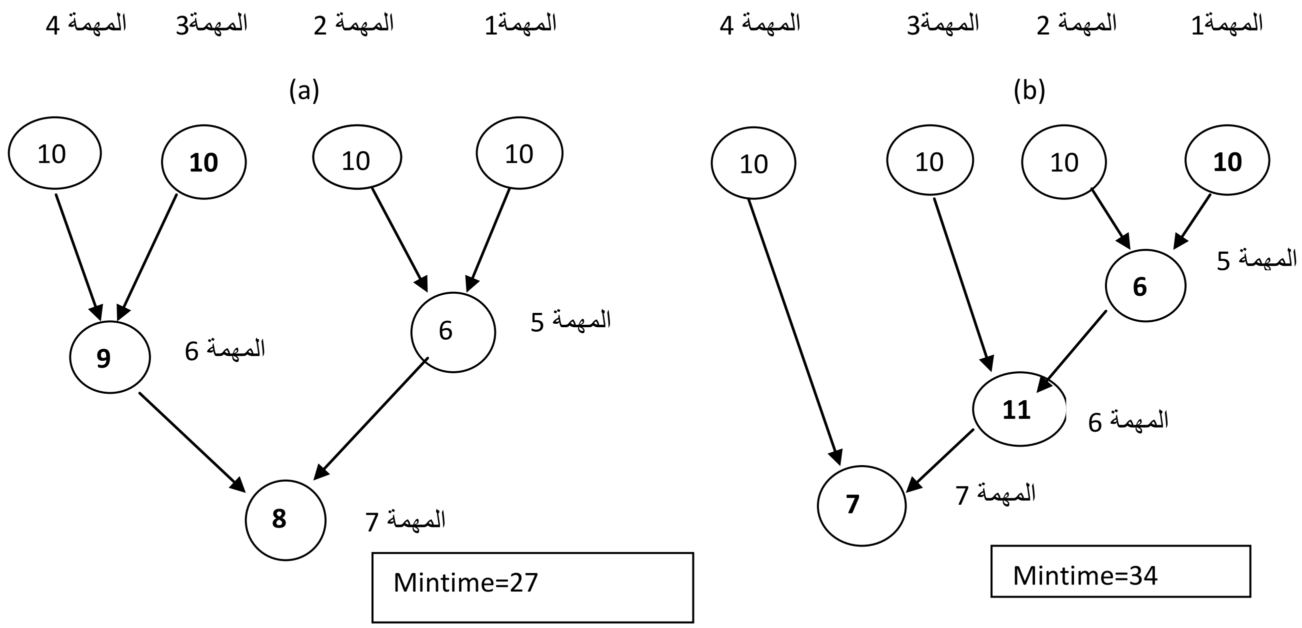
فعلى سبيل المثال التقسيم لمسألة ضرب مصفوفة بشعاع الوارد في الشكل (3-1) له تقسيم حبوبي صغير و درجة تزامن عالية. أما التقسيم لنفس المسألة في الشكل (3-4) له تقسيم حبوبي كبير و درجة تزامن منخفضة.

تعتمد درجة التزامن أيضا على شكل مخطط التبعية و التقسيم الحبوبي ذاته. و بشكل عام ، ليس هناك ضمان لتماثلها في درجة التزامن. فعلى سبيل المثال يعتبر الشكل (3-5) تجريدا لمخططي التبعية في الشكلين ((3-3) و (3-2)) على التوالي ، و العدد المكتوب بداخل كل عقدة يمثل كمية العمل المطلوب لإكمال مهمة المسندة لهذه العقدة.

إن معدل درجة التزامن لمخطط التبعية الموضح في الشكل (3-a.5) هو 2.33 ، و في الشكل (3-b.5) هو 1.88 ، مع أن كلا المخططان يعتمدان نفس التقسيم .

[9] Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama: ، مصدر سبق ذكره ، صص10-12

الشكل (3–5): تجريد لمخططي التبعية للشكلين (3–3) و (3–2)



المصدر1: : **Introduction to parallel Algorithms and Parallel Program design**", University of Oregon , p.25

المصدر2: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, المصدر سبق ذكره، ص.10

يتميز بيان التبعية على أنه يحدد معدل درجة التزامن لأي تقسيم حبوبي معطى عن طريق **المسار الحرج** ، نشير للعقد التي ليس لها أضلاع داخلة إليها بعقد البداية ، أما العقد التي لا يخرج منها أضلاع فنشير لها بعقد النهاية. و على هذا **فالمسار الحرج** هو أطول خط يصل بين أي زوجين من عقد البداية و النهاية . و أما المجموع لكمية العمل للعقد الواقعة على المسار الحرج يعرف بطول المسار الحرج ، بحيث أن كمية العقدة هي كمية العمل للمهمة المطابقة لهذه العقدة. أما نسبة إجمالي كمية العمل للمسار الحرج فتُعرَف **بمعدل درجة التزامن**. و لذلك فالمسار الحرج الأقصر يؤدي إلى درجة تزامن أعلى. فعلى سبيل المثال ، طول المسار الحرج لمخطط التبعية الموضح في الشكل (a.5–3) هو 27، أما للشكل (b.5–3) فالطول هو 34 ، و نظرا لأن مجموع كمية العمل اللازمة لحل المسألة باستخدام أسلوبي التقسيم هو 63 و 64 على التوالي فإن معدل **درجة التزامن لمخططي التبعية** هو 2.33 أي

(27/63) و 1.88 أي(34/64)على التوالي [10]. إن لمخطط إسناد المهام للمعالجات أهمية كبيرة في نجاعة الخوارزميات المتوازية. يتم إعداده من خلال بيان التبعية وبيان التفاعلات بين المهام إذ أنها تتشارك فيما بينها مدخلات و مخرجات و بيانات وسيطة . ويستعمل بيان التبعية للمهام للتأكد و ضمان أن العمل موزع بين جميع المعالجات ، و أما بيان تفاعل المهام فيستعمل للحصول على أدنى التفاعلات بين المعالجات وهذا بإسناد المهام التي بينها تفاعلات كثيفة لنفس المعالج. و بالتالي ففي كل الأحوال ، يهدف إعداد مخطط إسناد المهام إلى تخفيض زمن تنفيذ الخوارزمي المتوازي [11]. تتميز التفاعلات بين المهام بكونها منتظمة أو غير منتظمة. فالتفاعلات المنتظمة هي التي يمكن وضع مخطط لها. أما التفاعلات غير المنتظمة فهي التي ليس لها طوبولوجيا معروفة. تكون هذه التفاعلات في القراءة أو في القراءة و الكتابة. فالمهام يمكنها القراءة و الكتابة في عناصر معطيات مهام أخرى. فبصفة عامة التفاعلات في القراءة و الكتابة تكون أصعب في البرمجة و تتطلب أوامر مزامنة إضافية . و يمكن للتفاعلات بين المهام أن تكون ذات إتجاه واحد أو ذات إتجاهين. فالأول التفاعل يتم بين مهمة و أخرى ، بينما في الثاني فالتفعل متبادل بين مهمتين [12].

فقد يحصل هناك تفاعل فيما بين المهام التي تظهر مستقلة في مخطط التبعية. فمثلا... في التقسيم لمسألة ضرب مصفوفة بشعاع، و على الرغم من أن جميع المهام مستقلة عن بعضها البعض ، إلا أن جميع هذه المهام تتطلب الوصول إلى الشعاع b لأدائها. و نظرا لأن هناك نسخة واحدة من الشعاع b، فعلى المهام أن تُرسل و تستقبل الرسائل من الجميع لكي تصل إلى الشعاع b في نموذج الذاكرة المشتركة.

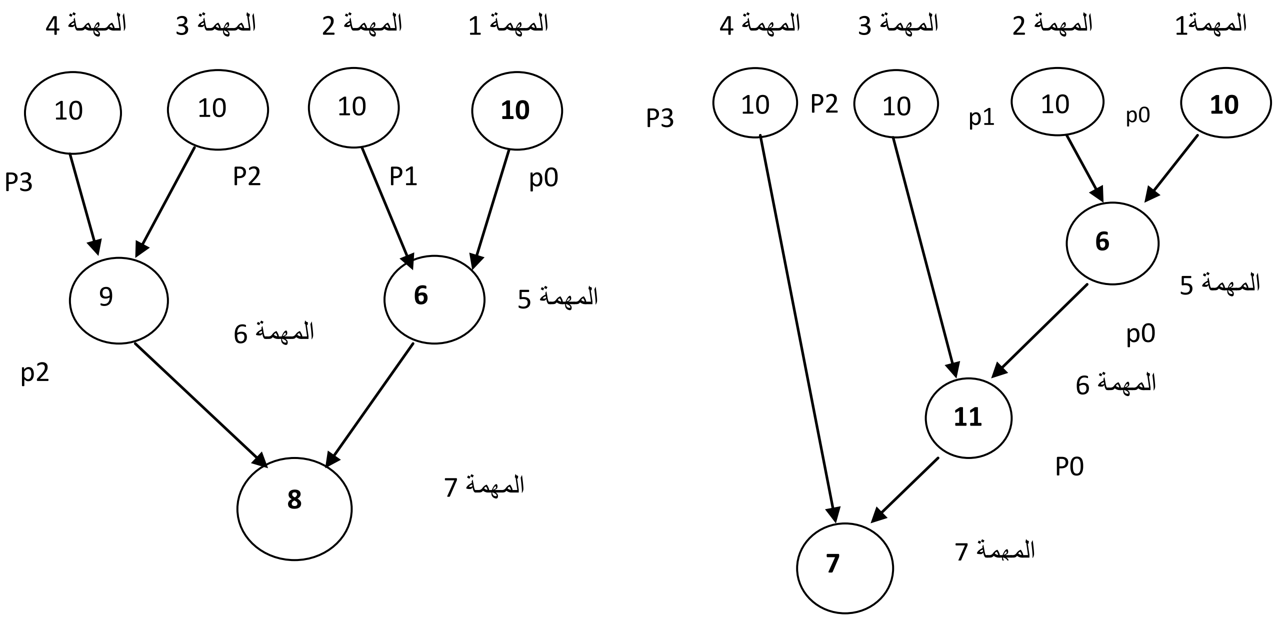
---

[10] : : Introduction to parallel Algorithms and Parallel Program design", University of Oregon , p.25.

[11]: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama ، مصدر سبق ذكره ،صص18-19

[12]: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama، مصدر سبق ذكره، صص57-58

الشكل (3-6): عملية الاسناد لمخطط المهام في شكل (3-5) إلى أربعة إجرائيات (Pi(i=1,4.



المصدر1: : **Introduction to parallel Algorithms and Parallel Program design**", University of Oregon , p.26

المصدر2: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama، المصدر سبق ذكره، ص10.

3-3 تقنيات التقسيم: يعتمد في التقسيم على تقنيات أساسية وهي[13]:

- **التقسيم البسيط(simple decomposition):** يتمثل في تقسيم المسألة إلى أجزاء مستقلة؛ كل جزء منها يمكن تنفيذه على معالج مختلف.
- **التقسيم بالتواصل(decomposition with communication):** في العديد من المسائل، يتطلب التواصل بين المعالجات قصد تبادل النتائج. و حل المسألة يتم في مرحلتين.

[13] : Daouda Traoré, "**Algorithmes parallèles auto-adaptatifs et application**", Institut polytechnique de Grenoble, 2008, pp.25-28.

في المرحلة الأولى، يتم تقسيم المسألة إلى أجزاء وكل جزء ينفذ على معالج مختلف. في المرحلة الثانية، يتم التواصل بين المعالجات لضم نتائجهم قصد انهاء حل المسألة. في الكثير من الأحيان يكون هذا التقسيم سواء البسيط أو بالتواصل حسب عدد المعالجات المستعملة. و هناك من الباحثين من ربط طريقة التقسيم بنوع المسألة المراد تقسيمها و ليس هناك وصفة وحيدة لجميع المسائل. ومن أشهر هذه التقنيات نذكر: التقسيم التراجعي ، تقسيم المعطيات ، التقسيم الإستكشافي و التقسيم التخميني [14].

1.3.3 التقسيم التراجعي (Recursive Decomposition) :

التقسيم التراجعي فهو عادة يتكيف مع المسائل من نوع فرق–تسد. وفيه تقسم المسألة إلى مسائل فرعية ناتجة عن التقسيم التراجعي. بمعنى كل واحدة من هذه المسائل الفرعية تُحل أيضا بتكرار تقسيمها إلى مسائل فرعية في نفس الوقت ثم تتُبع بنتائجها مجتمعة.

مثال (3–3): الفرز السريع (Quicksort):

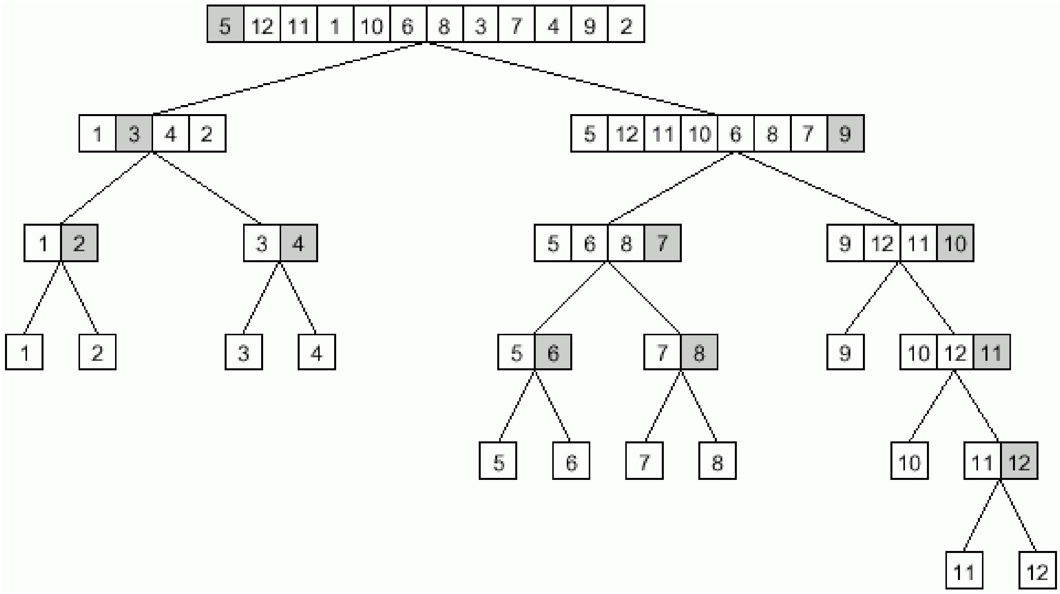
الترتيب أو الفرز السريع هي طريقة من نوع فرق–تسد، تقوم بترتيب قائمة عناصرT . تقسم إلى قائمتين جزئيتين T1 و T2 و التي تحتوي على التوالي على أصغر و أكبر العناصر بالنسبة لعنصر محوري، ثم بعد ذلك يتم تطبيق الخوارزمي تراجعيا على القائمتين T1 و T2. و كل من القائمتين الفرعيتين T1 و T2 يتم فرزها بواسطة الاستدعاء تراجعيا لخوارزمية الفرز السريع Quicksort. و كل استدعاء من هذه الاستدعاءات التراجعية يؤدي إلى تقسيم إضافي للقائمتين. تستعمل العديد من الطرق لتقسيم القائمة، و لكن المبدأ العام يتمثل في استعمال عنصر معين يسمى المحور(pivot) كقيمة تقسيم. يمكن اختيار العنصر الأول في القائمة مثلا، كما يمكن اختياره عشوائيا من القائمة. للعنصر المحور أهمية في تسريع عملية الترتيب ، لذا فاختياره أساسي. يعتبر خوارزمي الفرز السريع داخلي و في نفس الموضع في الذاكرة.[15]

يوضح الشكل (3–7) هذه المسألة مع فرز 12 عدد ، و يلاحظ أن الاستدعاء التراجعي لا يتوقف إلا عندما تحتوي كل سلسلة فرعية على عنصر وحيد فقط.

[14]: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama ، مصدر سبق ذكره ص21،ص27

[15]: Fréderic Vivien , "Algorithme Avancé", 2002, p.33

الشكل (3−7): مخطط التبعية للفرز السريع و القائم على التقسيم التراجعي لتقسيم سلسلة من 12 عدد.



المصدر1: "**Introduction to parallel Algorithms and Parallel Program design**", University of Oregon , p.34

المصدر2: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, المصدر سبق ذكره، ص.23

في الشكل (3−7) قد عرفنا المهمة بأنها القيام بتقسيم سلسلة فرعية معطاة. و على هذا فإن الشكل (3−7) يوضح أيضا مخطط المهمة الخاص بالمسألة. ففي البداية هناك سلسلة واحدة (جذر الشجرة). و يمكننا أن نستخدم عملية واحدة لتقسيمها ، و عند اكتمال مهمة الجذر فإنه ينتج عنها اثنتين من السلاسل الفرعية (A0 و A1 متوافقتين مع العقدتين في المستوى الأول من الشجرة) و كل منهما يمكن أن يُقسم بالتوازي ، و بنفس الطريقة يستمر التزامن في الزيادة كلما نزلنا إلى أسفل الشجرة.

في بعض الأحيان يمكن القيام بإعادة هيكلة العملية الحسابية و ذلك لجعلها قابلة للتقسيم العودي حتى لو كانت الخوارزمية المستخدمة للمسألة ليست من نوع (فَرّق− تَسُد). فعلى سبيل المثال لفرض أننا نريد إيجاد العنصر الأصغر في سلسلة غير مرتبة A مكونة من n عنصر. تقوم الخوارزمية التسلسلية لحل هذه المسألة بمقارنة كل عناصر السلسلة A ، و في كل خطوة تقوم بتسجيل أصغر عنصر موجود حتى الآن ، كما هو موضح في الخوارزمية 3−1. و من السهل إدراك أن هذه الخوارزمية ليست تزامنية.

الخوارزمية (3−1): برنامج تسلسلي لإيجاد العدد الأصغر في قائمة أعداد A بطول n.

```
Procedure SERIAL−MIN (A ,  n).
Begin
Min= A [0] ;
For i := 1 to n−1 do
        If (A[i] < min) min := A[i] ;
Endfor ;
Return min ;
End SERIAL –MIN
```

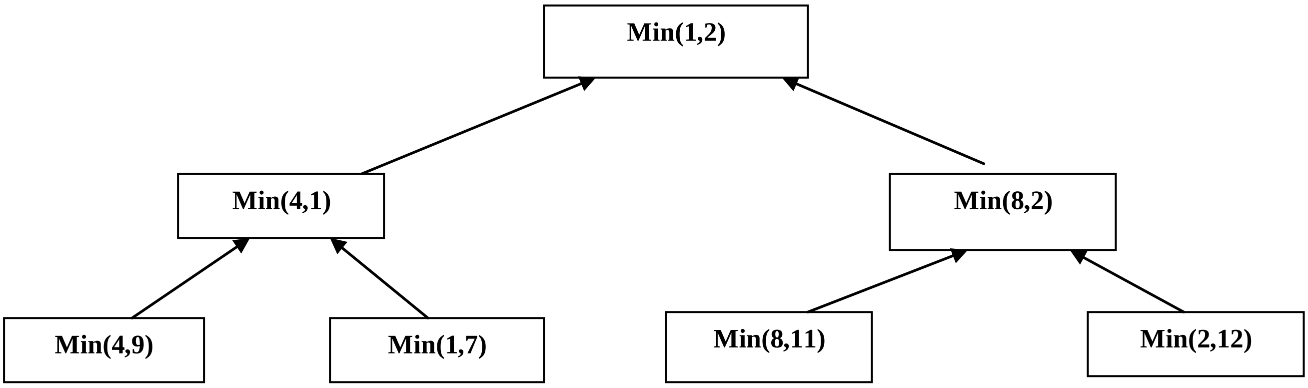
المصدر: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama، المصدر سبق ذكره، ص.24

حين نُعيد هيكلة هذه الخوارزمية لكي نجعلها من النوع "فَرّق− تَسُد" ، فإنه يمكن لنا استخدام التقسيم التراجعي كي نجعل منها خوارزمية متزامنة.

الخوارزمية (3−2) هي من النمط "فَرّق− تَسُد" و هي من أجل إيجاد العنصر الأصغر في مصفوفة ، و في هذه الخوارزمية نقوم بتقسيم السلسة A إلى سلسلتين فرعيتين لهما الحجم (n/2)، و من ثم نقوم بإيجاد العنصر الأصغر لكل واحدة من السلستين و ذلك باستخدام الاستدعاء العودي. و العنصر الأصغر الكلي يوجد بانتقاء أصغر عنصر في هذين السلستين. يتوقف الاستدعاء العودي فقط عندما يتبقى عنصر واحد في السلسلة. بعد أن أعدنا هيكلة الخوارزمية التسلسلية بهذا الأسلوب فإنه يكون من السهل رسم المخطط على المهمة لهذه المسألة.

لتوضيح ذلك نأخذ سلسلة الاعداد التالية :[4,9,1,7,8,11,2,12] ، نقوم بعد ذلك بتقسيمها إلى أربع أزواج و إيجاد أصغر عنصر في كل زوج لتتحصل على قائمة من زوجين فقط و بنفس الطريقة و تراجعيا نعاود العملية لنحصل في الأخير على العنصر الأصغر و يكون في جذر الشجرة كما يوضحه الشكل (3−8) حيث أن كل عقدة في الشجرة تمثل مهمة لإيجاد العدد الأصغر من عددين.

الشكل (3−8): مخطط التبعية لإيجاد العدد الأصغر للسلسلة أعداد



المصدر: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama، المصدر سبق ذكره ص 26

الخوارزمية(3−2): برنامج تراجعي لإيجاد العدد الأصغر Min من بين عناصر A المكونة من n عددا.

```
Procedure RECURSIVE – MIN (A , n)
Begin
If (n=1) then
   Min := A [0] ;
 Else
        Lmin := RECURSIVE – MIN (A , n/2) ;  //Left min
       Rmin := RECURSIVE – MIN ( (A [n/2] ) , n−n/2) ; // Right min
       If (lmin <rmin) then
           Min := lmin ;
   else
       Min := rmin ;
   Endelse ;
Endelse ;
Return min ;
End RECURSIVE – MIN
```

المصدر: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama، المصدر سبق ذكره، ص.10

3.2 3. تقسيم المعطيات (Data Decomposition) :

و يتمثل هذا التقسيم في تحديد المعطيات التي ستجرى عليها الحسابات. تقسم هذه المعطيات بين مختلف المهام مما يؤدي بالتالي إلى تقسيم المسألة. و يوجد عدة طرق لتقسيم المعطيات سواء على مستوى المدخلات أو على مستوى المخرجات. فغالبا ما يمكن حساب كل عنصر في المخرج بمعزل عن العناصر الأخرى و لكن فقط من خلال المدخل. و يعتبر هذا التقسيم أسلوباً فعّالاً و شائعاً يستخدم للحصول على التزامن في الخوارزميات التي تنفذ على المعطيات كبيرة الحجم وعادة ما تكون المهام بها بسيطة و متشابهة [16]. و في المثال (3-4) سنعرض لعمليات ضرب مصفوفة لإيضاح التقسيم المعتمد على تجزئ البيانات المخرجة.

المثال (3-4): ضرب المصفوفات المربعة

بفرض أننا نريد إجراء عملية الضرب على المصفوفتين (A و B) و كلاهما من الحجم nxn و سنقوم بوضع الناتج في المصفوفة C. في الشكل (3-3) توضيح لتقسيم هذه المسألة إلى أربع مهام. حيث تم اعتبار أن كل مصفوفة مركبة من أربع كتل (أو مصفوفات جزئية) تحدد هذه الكتل بواسطة تقسيم كل بعد من المصفوفة إلى نصفين (و بذلك سينتج لدينا أربع كتل داخل المصفوفة). و المصفوفات الجزئية الأربع للمصفوفة C (من الحجم n/2xn/2) يتم حسابها مستقلة باستخدام أربع مهام كمجموع لحواصل الضرب الموافق للمصفوفات الجزئية الموجودة في A و B .

[16]: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama: ، مصدر سبق ذكره، ص ص27-28

الشكل رقم (3−9): (a) تجزئة مصفوفات المدخلات و المخرجات إلى مصفوفات جزئية بحجم 2x2.
(b) التقسيم لمسألة ضرب المصفوفات إلى أربع مهام اعتمادا على تجزيء المصفوفات الوارد في (a)

$$\begin{pmatrix} A_{1.1} & A_{1.2} \\ A_{2.1} & A_{2.2} \end{pmatrix} + \begin{pmatrix} B_{1.1} & B_{1.2} \\ B_{2.1} & B_{2.2} \end{pmatrix} \rightarrow \begin{pmatrix} C_{1.1} & C_{1.2} \\ C_{2.1} & C_{2.2} \end{pmatrix}$$

(a)

Task 1 : $C_{1.1} = A_{1.1}\, B_{1.1} + A_{1.2}\, B_{2.1}$

Task 2 : $C_{1.2} = A_{1.1}\, B_{1.2} + A_{1.2}\, B_{2.2}$

Task 3 : $C_{2.1} = A_{2.1}\, B_{1.1} + A_{2.2}\, B_{2.1}$

Task 4 : $C_{2.2} = A_{2.1}\, B_{1.2} + A_{2.2}\, B_{2.2}$

(b)

المصدر: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama، المصدر سبق ذكره، ص.29

إن التقسيم الموضح في الشكل (3−9) قائم على تجزيء مصفوفة المخرج C إلى أربع مصفوفات جزئية و تسند كل واحدة منها إلى مهمة تقوم بحسابها.

يمكن لتقسيم بيانات المخرج أن يصدر عنه أكثر من تقسيم للمهام. فعلى سبيل المثال الشكل (3−10) يوضح تقسيمين آخرين لضرب المصفوفات ، كل واحد إلى ثماني مهام ، و هذان التقسيمان مماثلان لنفس تقسيم البيانات الموجودة في الشكل السابق (3−9.a)[17]. و تنفيذ كل منهما يؤدي لنفس النتيجة و هي حساب المصفوفة C بالتوازي.

[17]: : Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama، المصدر سبق ذكره، ص.30

الشكل (3–10): تقسيم عملية ضرب المصفوفة إلى ثمانية مهام.

التقسيم I	التقسيم II
Task 1: $C_{1,1} = A_{1,1}B_{1,1}$	Task 1: $C_{1,1} = A_{1,1}B_{1,1}$
Task 2: $C_{1,1} = C_{1,1} + A_{1,2}B_{2,1}$	Task 2: $C_{1,1} = C_{1,1} + A_{1,2}B_{2,1}$
Task 3: $C_{1,2} = A_{1,1}B_{1,2}$	Task 3: $C_{1,2} = A_{1,2}B_{2,2}$
Task 4: $C_{1,2} = C_{1,2} + A_{1,2}B_{2,2}$	Task 4: $C_{1,2} = C_{1,2} + A_{1,1}B_{1,2}$
Task 5: $C_{2,1} = A_{2,1}B_{1,1}$	Task 5: $C_{2,1} = A_{2,2}B_{2,1}$
Task 6: $C_{2,1} = C_{2,1} + A_{2,2}B_{2,1}$	Task 6: $C_{2,1} = C_{2,1} + A_{2,1}B_{1,1}$
Task 7: $C_{2,2} = A_{2,1}B_{1,2}$	Task 7: $C_{2,2} = A_{2,1}B_{1,2}$
Task 8: $C_{2,2} = C_{2,2} + A_{2,2}B_{2,2}$	Task 8: $C_{2,2} = C_{2,2} + A_{2,2}B_{2,2}$

المصدر: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama، المصدر سبق ذكره، ص.30.

3.3.3 التقسيم الاستكشافي (Exploratory Decomposition)

في الكثير من الأحيان يتماشى تقسيم المسألة مع طريقة تنفيذها . و تتمحور هذه المسائل في البحث عن فضاء الحلول لها. كما تشمل هذه الفيئة بعض مسائل الأمثلية كالبرمجة الصحيحة مثلا.ففي التقسيم الاستكشافي نقوم بتقسيم فضاء البحث إلى أجزاء صغيرة ، و يتم البحث في كل الأجزاء بشكل متزامن إلى أن يتم إيجاد الحل المطلوب[18].

[18]: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama، المصدر سبق ذكره، ص.41.

4.3.3 التقسيم التخميني (speculative decomposition)

تكون التبعية بين المهام غير معروفة مسبقا و بالتالي يصعب تحديدها في هذه المسائل. يوجد مقاربتين لمعالجة هذه المسائل. المقاربة التحفظية و المقاربة التفائلية. ففي المقاربة التحفظية يتم تحديد المهام المستقلة فقط عندما لا يكون بينها تبعية. أما المقاربة التفائلية فتوضع مخطط لهذه المهام و لو أدى تنفيذها لنتائج مغلوطة. كما تسمح هذه المقاربة بالرجوع للوراء في حالة الخطأ. يستخدم التقسيم التخميني في البرامج التي قد تأخذ تفرع واحد من بين عدة تفرعات حسابية اعتمادا على النتائج لعمليات حسابية سابقة. و يمكن للتسريع الذي ينتج عن التقسيم التخميني أن يكون كبيرا إذا كان هنالك مراحل تخمينية متعددة[19].

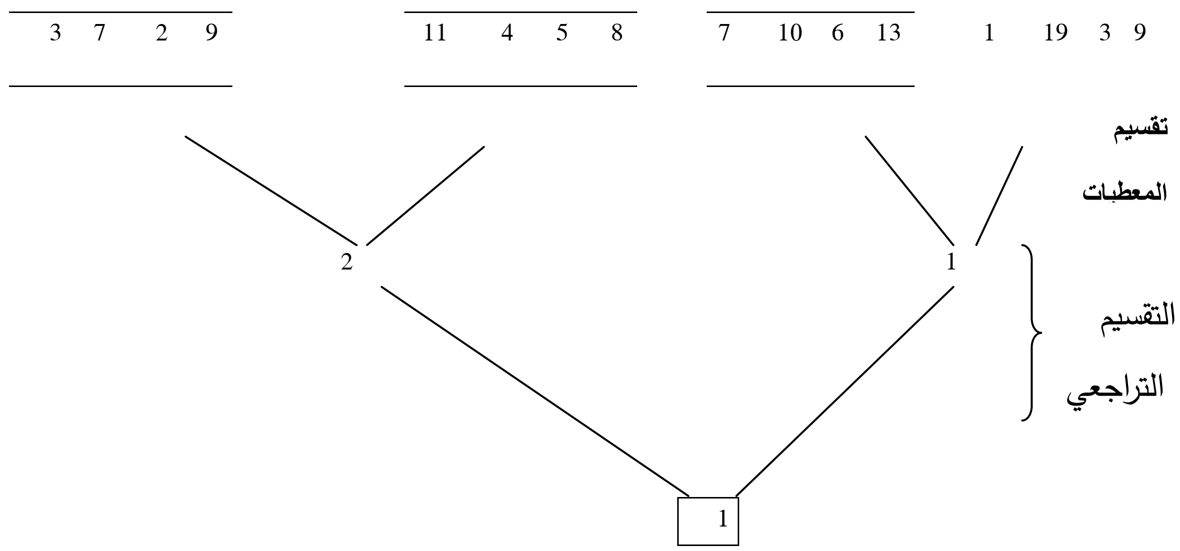
5.3.3 التقسيم المختلط (hybrid decomposition)

يكون خليط تقنيات التقسيم في أغلب الأحيان ضروري و محبذ لتقسيم المسائل. و عادة ما تكون هذه المسائل مكونة من عمليات مركبة و في مراحل يكون من الأفضل فيها استعمال تقنيات تقسيم مختلطة في كل مرحلة من مراحل الحساب. و هو ما يبينه مثال إيجاد أصغر عنصر لقائمة أعداد مكونة من ستة عشرة عددا . إذ قسمت إل أربع أجزاء متساوية و أسند كل جزء إلى مهمة تتكلف بالبحث عن أصغر عنصر بها ثم بعد ذلك استخلاص العنصر الأصغر النهائي. استعمل في هذا تقسيم المعطيات و التقسيم التراجعي و هو ما يوضحه الشكل(3−11) .[20]

[19]: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, ، المصدر سبق ذكره، ص45

[20]: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama: ،مصدر سبق ذكره ، ص48

الشكل (3-11): التقسيم المختلط لإيجاد العدد الأصغر لقائمة من 16 عددا باستخدام أربعة مهام



المصدر: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama، المصدر سبق ذكره، ص.48

6.3.3 القواعد الأساسية للتوازي في الخوارزميات

هناك عدة قواعد ترتكز عليها عملية التحويل للتوازي في البرامج الحاسوبية . نتطرق إلى أهمها فيما يلي[21]:

1- توازي الكتلة القاعدية(parallelization of a base block)

تعرف الكتلة القاعدية على أنها سلسلة تعليمات بدون مراقبة. و إذا تم تنفيذ تعليمة منها، تتبعها التعليمات الأخرى. يكون شكل الكتلة في لغة البرمجة كالتالي:

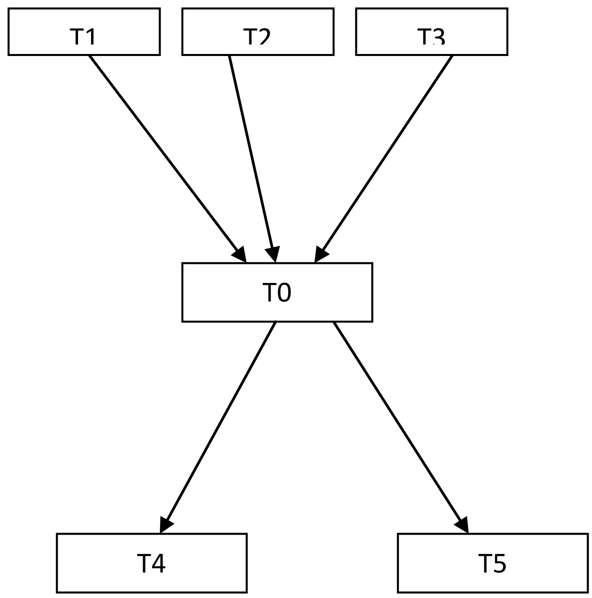
begin s1; s2;…..;sn end أو **{s1; s2; …..;sn}** أين si تمثل سلسلة التعليمات.

[21] :Paul Feautrier , Méthode élémentaire de parallélisation , ENS de Lyon , november (2008) p.3

أ- **مخطط التبعية:** يمثل مخطط التبعية البرنامج، وكل تعليمة إجرائية (processus)، وأضلاع المخطط عمليات التزامن. لا تكون لها أهمية عملية إلا عندما تكون مدة التعليمة أطول بالنسبة لعملية التزامن (synchronisation).

ب- **التوازي على مستوى الهيكل(shell):** يمكن توضيح عملية التزامن باستعمال الشكل الموالي:

الشكل(3-12): التوازي على مستوى الهيكل(shell)



المصدر: Paul Feautrier ، مصدر سبق ذكره ، ص7.

من خلال هذا الشكل(3-12) نرى مثلا أن المهمة T0 لا يمكنها الانطلاق في التنفيذ إلا بعد انتهاء المهام T1, T2, T3 و كذلك المهمة T4 و T5 لا يمكنهما انطلاق في التنفيذ إلا بعد انتهاء المهمة T0.

ج- **الجدولة(scheduling):** يرتكز مبدأ الجدولة على ما يلي[22]:

- نفترض أننا نعرف مدة تنفيذ كل تعليمة (مثلا: نفس المدة لكل تعليمة).
- نعرف كذلك عدد المعالجات(processors).
- نختار لكل تعليمة مدة التنفيذ و معالج تحت القيود:
  - ✓ الإستجابة للتبعية

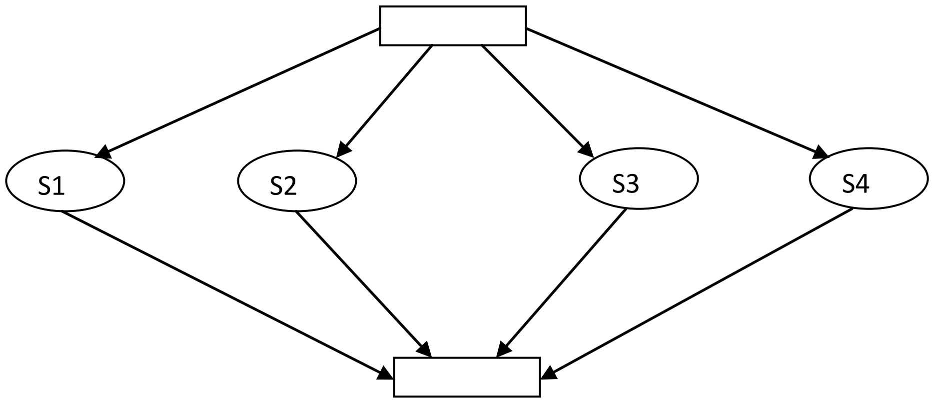
[22] Paul Feautrier: ، مصدر سبق ذكره ، ص ص7-8.

- ✓ إستعمال المعالجات المتاحة فقط
- ✓ يمكننا بعد ذلك تنفيذ البرنامج من خلال جدول مواعيد(timetable) الذي يقوم بإطلاق تنفيذ كل تعليمة في الوقت المراد ، أو كتابة برنامج مناسب لجدول المواعيد.

د- سلسلة تجميع/موازاة (compilation series/parallel)

تمنح العديد من الأنظمة معامل (operator) خاص بكل منها و بترميزات خاصة بها لإعلان التوازي كما يوضحه الشكل(3-13) التالي:

الشكل(3-13): سلسلة تجميع/موازاة (compilation series/parrallel)



المصدر: Paul Feautrier ، **مصدر سبق ذكره ، ص. 21**

من خلال الشكل(3-13) يتضح أن المهام (s1 , s2, s3, s4) تنفذ بالتوازي أي في نفس الوقت.

و تختلف هذه الرموز من لغة برمجة متوازية لأخرى كما هو موضح من خلال التعليمات التالية [23]:

S1//S2// S3 // S4

cobeging S1 ; S2 ; S3 ; S4 coend (Algol)

{| S1 ; S2 ; S3 ; S4 |}   (Earth C)

PARALLEL SECTIONS   (Open MP)

  S1

PARALLEL

  S2

---

[23]: Paul Feautrier ، مصدر سبق ذكره ، ص21

……

…….

END PARALLEL SECTION

فمثلا في التعليمة الأولى ، أستعمل الرمز // لإعلان التوازي للعمليات (s1, s2, s3, s4) و في التعليمة الثانية في لغة البرمجة Algol تحاط العمليات التي تجرى بالتوازي(s1, s2,s3, s4) بين الكلمتين المفتاحيتين cobegin و coend و هكذا لبقية لغات البرمجة المتوازية.

هـ- كيفية معالجة الإختبار في التوازي(treatment tests)

يتم معالجة الإختبار في التوازي بإتباع الخطوات التالية :

أولا نقوم بتجميع فرعي للاختبار.

ثانيا تحويل تابعيات التحكم إلى تابعيات عادية من خلال if conversion.

If (p)		**boolean beta ;**
S1 ;		**beta = (p==0)**
else	⟹	**if (beta) S1 ;**
S2 ;		**if (!beta) S2 ;**

ثالثا ينفذ بالتوازي الاختبار و الفرعين ، ثم تثبت نتائج الفرع المختار في الاختبار. في البرنامج المتسلسل يتم اختبار الشرط P . فإذا كان الشرط محقق تنفذ التعليمة S1 و إلا تنفذ التعليمة S2. في البرنامج المتوازي نستعمل متغير منطقي إضافي**beta** تخزن فيه نتيجة الشرط P (صحيح/خطأ) و تنفذ التعليمة IF على فرعين. الفرع الأول على الشرط beta (صحيحة) و الثاني على الشرط المعاكس !beta .

2- التوازي في الحلقات (loop parallelization): إن الاستغلال الأمثل لأعشاش الحلقات (الحلقات التكرارية المتوازية) يتضمن استغلال إمكانات بنية الحاسوب من حيث تعدد وحدات الحساب قصد إحلال توازي المعالجة. و تتمثل الطريقة في إبراز المعالجة التكرارية و التي يمكن أن تؤدى بالتوازي

و في نفس الوقت ، ثم بعد ذلك إسناد كل منها إلى وحدة حساب معينة. يعتمد التوازي بشكل كبير على ترتيب تبعية البيانات في أعشاش الحلقات. إن غياب تابعيات البيانات بين التكرارات يسهل عملية التوازي. تمثل تبعية البيانات في نقل هذه الأخيرة بين تعليمتين. و يمكن تصنيفها وفقا لترتيب عمليات القراءة و الكتابة في المتغيرات المستخدمة. نذكر ثلاث أنواع منها:

النوع الأول، يتمثل في تبعية التدفق ، هنا نجد التعليمة الأولى تكتب في متغير و تليها قراءة تعليمة أخرى في نفس المتغير.

النوع الثاني، هو الاستقلالية و فيه نجد تعليمة تقوم بقراءة محتوى متغير قبل أن تكتب فيه تعليمة أخرى.

النوع الثالث و هو التبعية في الاخراج و يتمثل في أن تعليمتين تكتبان في نفس المتغير ، علاوة على ذلك ، يمكن تصنيف تابعيات البيانات في أعشاش الحلقات وفقا الانتماء لتعليمات الإنتاج أو الاستهلاك في التكرارات. و يمكن أن نميز بين نوعين منها. النوع الأول و هو تبعية بيانات التكرار المتبادل و هو عبارة عن تبعية للبيانات بين تعليمتين تنتميان لتكرارين مختلفين . أما النوع الثاني فيتمثل في تبعية بيانات التكرار الداخلي و هي تبعية للبيانات بين تعليمتين تنتميان لنفس التكرار[24]. بالإضافة إلى ما سبق يمكن القول أن الحلقة المستقلة هي الحلقة التي لا تحتوي على متغيرات وسيطة أو متغيرات القراءة و الكتابة، و التي يمكن تنفيذ التكرارات بها في أي ترتيب. أما الحلقة التابعة(المرتبطة) ، فهي الحلقة التي تحتوي متغيرات وسيطة أو متغيرات إدخال و إخراج حيث تحوي المتغيرات في التكرار ارتباطات في مؤشرها من جانبي العبارة ، في اليسار و اليمين. و هو ما سنوضحه في الأمثلة الموالية:

المثال الأول: نفترض لدينا الحلقة التكرارية for التالية:

```
for i= 1: l  do
   a(i) = a(i) + b(i)
end for
```

هنا في المثال الحلقة for مرتبطة ، لكن المتغير a لها نفس المؤشر i في يسار و يمين عبارة التعيين

[24] :Yaroub ELLOUMI, "Parrallélisme des nids de boucles pour l'optimisation du temps d'execution", pp22-23

$a(i) = a(i) + b(i)$ ، لذا يمكن تنفيذ كل تكرار باستقلالية عن التكرارات الأخرى.

المثال الثاني: نفترض لدينا الحلقة for الموالية:

```
for i= 1: I do
  a(i) = a(i−1) + b(i)
endfo
```

ف هذا المثال بالمقابل، الحلقة for مرتبطة كونها تحوي متغير وسيطي a ، و هناك كذلك ارتباط في المؤشر i في جانبي عبارة التعيين $a(i) = a(i-1) + b(i)$ و بالتالي لا يمكن أداء التكرار i دون حساب التكرار $(i-1)$. هذا النوع من الحلقات يمكن تنفيذها بالتسلسل عل حاسب مكون من معالج واحد أو من عدة معالجات. نشير في الأخير إلى أن متغير الإدخال أو القراءة هو المتغير الذي يكون في يمين العبارة فقط بينما متغير الإخراج أو الكتابة فيكون في يسار العبارة فقط و الوسيط يكون في يسار و يمين العبارة [25].

إن وجود التوازي في التعليمات أو الحلقات التكرارية يتطلب وجود الشروط التالية [26]:

- لا يمكن تحديد ترتيب التنفيذ في الحلقة المتوازية

```
// for(i=0 ; i<n ; i++)
            S ;
```

- إذا كان لدينا P معالج، كل معالج يمكنه تنفيذ التكرارات P ب P.
- يمكن كذلك للحلقة أن تقسم إلى كتل متساوية من حجم P .
- لا يوجد ترتيب للحلقة المتوازية.
- يمكن إيجاد التوازي في الحلقة، شريطة عدم وجود أي تبعية بين تكراراتها.
- لا يجب أن تكون تبعية من عمق $p-1$ في حلقة من الرتبة p.

---

[25]: Fayez Gebali ، مصدر سبق ذكره صص134-135
[26]: Paul Feautrier ، مصدر سبق ذكره ، ص27

مثال(3−5): هل الحلقة على i متوازية؟

```
For (i=0 ; i<n ; i++)
  {
     Z : c[i] = 0. ;
       For (j=0 ; j<n ; j++ )
      M :   c[j] += a[i][j]* b[j] ;
}
```

مثال(3−6): حلقة يتم فيها تغيير عدد(متغير) تكون متسلسلة.

```
For (i=0 ; i<n ; i++) {
Z ; s=0. ;
     For (j=0 ; j<n ; j++)
M :     s + = a[i][j] * b[i] ;
C :  c[i] = s ;
}
```

مثال(3−7): تراكم الحسابات في نفس الحلقة يعيق عملية التوازي

```
For (i=0 ; i<n ; i++){
S : s += a[i] ;
A : b[i] = c[i] + d[i] ;
}
```

مثال(3−8): خوارزمية متسلسلة/متوازية

```
k=0 ;                      |
for (i=0 ; i<n ; i++){     |     for (i=0 ; i<n ; i++)
  A[k] = 0 ;               |       a[3*i] = 0 ;
   K += 3 ;                |
}                          |
```

الخوارزمي متوازي الخوارزمي متسلسل

3−تحويلات البرنامج (Program transformations)

إن الهدف من التحويلات في البرنامج هو لتحسينه من ناحية التنفيذ و التخزين، ولكن غالبا ما يؤدي إلى تحطيم التوازي فيه [27].

- التحويل الأول: هو تنفيذ نفس التعليمات ولكن في ترتيب مختلف.
- التحويل الثاني: نقوم بحساب نفس القيم ولكن تخزينها في الذاكرة يكون مختلفا.
- إذا قمنا بأي تحويل في برنامج، يجب أن يكون الإصدار الأول و الثاني متكافئتان.

```
For (i=………){               for (i=………)
  S1 ;         ========>     S1 ;
  S2 ;                      for (i=………)
}                             S2 ;
```

ملاحظة:− يمكن إجراء التفكيك و لكن شرط عدم وجود تبعية بين التعليمات S2 و S1.

وتكون الحلقة For غير قابلة للتفكيك إذا كانت هناك تبعية من S1 إلى S2 ومن S2 إلى S1.

[27] : Paul Feautrier ، مصدر سبق ذكره ، صص40-41

مثال(3−9): تفكيك الحلقة(bursting loops)

```
for(i=0 ; i<n ; i++) {

S : s += a[i] ;

A : b[i] = c[i] + d[i] ;

}
```

ملاحظة: − هناك تبعية ل S على نفسها، ولكن لا توجد من S نحو A.

− لا توجد تبعية ل A على نفسها. التفكيك إذن موجود، والحلقة على A متوازية.

وتكتب الحلقة كما يلي:

```
for (i =0 ; i<n ; i++ )

S : s += a[i] ;

// for (i=0 ; i<n ; i++)

A : b[i] = c[i] + d[i] ;
```

- دمج الحلقات(merging loops)
  - ✓ هي التحويل المعاكس لتفكيك الحلقة. عندما لا يوجد توازي في الحلقة، يكون الدمج أفضل بتوفير الاختبار.
  - ✓ تعليمة كثيرة التعقيد، يصعب جعلها متوازية لأنها تحتوي على تبعيات كثيرة.

```
For (i=0 ; i<n ; i++)

  s= s + a[i] * b[i] ;
```

✓ يمكن فصل عملية الضرب و الجمع من خلال إدخال متغير مؤقت tmp:

```
for (i=0 ; i<n ; i++) {
    tmp[i] = a[i] * b[i] ;
    s+= tmp[i] ;
}
```

✓ ثم بعد ذلك فك الحلقة و جعل عملية الضرب متوازية بشرط أن يكون المتغير المؤقت tmp جدول.

- تبديل الحلقات (loops permutation)

```
For (i=...........)                          for (j=...........)
    For (j=.........)  ————————→                for (i=.........)
        S ;                                          S ;
```

مثال(3−10): تبديل الحلقات (loops permutation)

```
For (i=0 ; i<n ; i++) {
Z : c[i] = 0. ;
      For (j=0 ; j<n ; j++)
M :         c[i] += a[i] [j] * b[j] ;
}
```

في الحلقة i المتوازية، تفكك الحلقة i ، ثم نقوم بتبديلها مع الحلقة j.

```
For (i=0 ; i<n ; i++)
```

```
Z :    c[i] = 0.
        For (j=0 ; j<n ; j++)
           // for (i =0 ; i<n ; i++)
M :              c[i] += a[i] [j]* b[j] ;
```

- الذاكرة و التوازي(memory and parallelism): نلاحظ أن هناك علاقة بين سعة الذاكرة ودرجة التوازي. إذا كان و لا بد من تنفيذ n عملية على التوازي و إذا كانت كل عملية ضرب لها نتيجة ، يجب n خلية من الذاكرة لتخزينها وبدون ذلك ستكون هناك تبعية. إذا كانت n خلية من الذاكرة غير معلنة في البرنامج، فالتوازي لا يمكن إستغلاله [28].

4.3 أمثلة الخوارزميات المتوازية:

نتطرق فيما يلي لبعض أمثلة الخوارزميات المتوازية ، و سوف تكون الطريقة العامة لعرض الخوارزمية هي: عرض الصيغة التسلسلية للخوارزمية ثم مناقشة كيفية جعلها متوازية . و نخص من بينها خوارزميات الفرز والتي تتمثل مهامها في ترتيب قائمة غير مرتبة و هذا بتبديل العناصر فيما بينها. هذا الترتيب يمكن أن يكون داخلي أو خارجي. فالترتيب الداخلي يتم في الذاكرة المركزية. أما الترتيب الخارجي فيتم بالاستعانة بالذاكرة الخارجية (القرص الصلب مثلا) [29].

1.4.3 خوارزمية الفرز الفقاعي (Bubble Sort).

تتمثل طريقة الفرز الفقاعي في مقارنة كل عنصر من القائمة مع العنصر الذي يليه و استبدالهما ان كان غير مرتبين. تكون القائمة مرتبة عندما لم يكن هناك تبديل لأي عنصر خلال التكرار [30]. لنفرض أن لدينا القائمة $<a_1,a_2,\ldots a_n>$ ، تقوم الخوارزمية أولا بإجراء $n-1$ عملية "مقارنة– و استبدال" في الترتيب التالي:(a_1,a_2) , (a_2,a_3) ,...., (a_{n-1},a_n) . بهذه الطريقة نزيح العنصر الأكبر إلى نهاية القائمة في حالة الترتيب التصاعدي أو العنصر الأصغر في حالة الترتيب التنازلي.

[28] : Paul Feautrier ، مصدر سبق ذكره ص ص47-48

[29] : **Introduction to parallel Algorithms and Parallel Program design"**, University of Oregon , pp. 45-46

[30] :Irene Guessarian, "Quelques Algorithmes Simples", 2012, p.8

إنه من الصعب جعل خوارزمية الفرز الفقاعي متوازية ، و يجب التفكير في كيفية أداء عمليات المقارنة – و الاستبدال أثناء كل مرحلة من الخوارزمية (السطرين 4 و 5 من خوارزمية (3-3). تقوم خوارزمية الفرز الفقاعي بمقارنة جميع الأزواج المتجاورة بالترتيب ؛ و لهذا السبب فهي بالدرجة الأولى خوارزمية تسلسلية. و في ما يلي سنعرض أحد أنواع الفرز الفقاعي و التي بالإمكان جعلها متوازية.

خوارزمية (3-3): خوارزمية الفرز الفقاعي التسلسلي

```
1. Procedure BUBBLE- SORT (n)
2. Begin
3.      For i := n-1 down to 1 do
4.           For j := 1 to i do
5.                Compare-exchange (a_j , a_j+1)
6. End BUBBLE- SORT
```

المصدر : Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, "**Introduction to parallel Computing:Sorting Algorithms** ", Addison Wesley, 2003, p.33

1.1.4.3 الإبدال الزوجي- الفردي (Odd-Even Transposition) :

خوارزمية "الإبدال الزوجي – الفردي" هي إحدى حالات الفرز الفقاعي و هي أكثر بطأ من الفرز الفقاعي التسلسلي كون كمية البيانات المتبادلة بين المعالجات كبيرة جدا و يمكن مقارنتها بعدد العمليات الحسابية المنفذة . تزداد كلفة الحسابات و عمليات تبادل المعطيات بين المعالجات مع ارتفاع عدد المعالجات [31]. يتم الفرز في خوارزمي الإبدال الفردي-الزوجي في مراحل فردية و زوجية.و يتطلب فرز n عنصر في n مرحلة (n زوجي) ، كل مرحلة تتطلب n/2 من عمليات المقارنة و الاستبدال . و هذه الخوارزمية تتناوب بين مرحلتين و هما المرحلة الفردية و المرحلة الزوجية . فإذا أردنا ترتيب السلسلة $<a_1,a_2,\ldots a_n>$. فخلال المرحلة الفردية سيتم مقارنة العناصر ذات الدليل الفردي مع ما يجاورها إلى اليمين ، فإذا لم يحققا شرط الترتيب فإنه يتم استبدال أماكنهما ؛ و بالتالي فالأزواج $((a_1,a_2)\ ,\ (a_3,a_4),\ldots,\ (a_{n-1},a_n))$ تقارن-و تستبدل . و نفس الشيء يتم عمله خلال المرحلة الزوجية ،إذ يتم مقارنة العناصر التي لها دليل زوجي مع ما يجاورها ناحية

[31] :GERGEL v. p. , "Introduction to parallel programing:Parallel methods for Sorting", p.15

اليمين ، فإذا لم يحققا شرط الترتيب فإنه يتم استبدال لأماكنهما و بالتالي فالأزواج (, (a_2,a_3) (a_4,a_5),........, (a_{n-2},a_{n-1})) يتم مقارنتها و استبدالها. و بعد n مرحلة فإنه السلسلة تكون قد رتبت . و يوضح المثال (3−10) عمل خوارزمية الإبدال الزوجي – الفردي التالي:

المثال (3−10): فرز 8 عناصر (n=8) باستخدام خوارزمية الإبدال الزوجي− الفردي ، خلال كل مرحلة هناك 8 عناصر يتم مقارنتها (n=8).

البداية : السلسلة غير مرتبة

المعالجات	P1	P2	P3	P4	P5	P6	P7	P8
المرحلة 1 (فردي)	3	2	3	8	5	6	4	1
المرحلة 2 (زوجي)	2	3	3	8	5	6	1	4
المرحلة 3 (فردي)	2	3	3	5	8	1	6	4
المرحلة 4 (زوجي)	2	3	3	5	1	8	4	6
المرحلة 5 (فردي)	2	3	3	1	5	4	8	6
المرحلة 6 (زوجي)	2	3	1	3	4	5	6	8
المرحلة 7 (فردي)	2	1	3	3	4	5	6	8
المرحلة 8 (زوجي)	1	2	3	3	4	5	6	8
	1	2	3	3	4	5	6	8

النهاية: السلسلة مرتبة

المصدر1: **Introduction to parallel Algorithms and Parallel Program design"**, University of Oregon , p.56

المصدر2: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, "**Introduction to parallel Computing:Sorting Algorithms**", Addison Wesley, 2003, p.36

الخوارزمية (3-4): الخوارزمية التسلسلية للإبدال الفردي- الزوجي

```
Procedure ODD- EVEN (n)
Begin
     For i := 1 to n do
     Begin
          If i is  odd then      // i%2==1 odd iteration
              For j := 0 to n/2-1 do
                   Compare-exchange (a2j+1, a2j+2) ;
          If i is even then      // i%2==0 even iteration
              For j := 1 to n/2-1 do
               Compare-exchange (a2j , a2j+1) ;
End for
End ODD- EVEN
```

المصدر1: **Introduction to parallel algorithm and parallel programs design**, university of Oregon, p.57

المصدر2: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, "**Introduction to parallel Computing:Sorting Algorithms**", Addison Wesley, 2003, p.35

الصيغة المتوازية لخوارزمية الإبدال الزوجي –الفردي:

نفترض في هذه الخوارزمية أن عدد الإجرائيات (المعالجات) هو نفسه عدد العناصر المراد ترتيبها سواء تصاعديا أو تنازليا، أي يساوي N عنصرا. فطريقة الفرز أو الترتيب الفردي-الزوجي و مع اعتبار أن كل معالج Pj يحتوي على عنصر واحد Xj (j=1, 2, ….,N). تتم عملية الفرز على مراحل. فخلال كل مرحلة من الخوارزمية يتم إجراء عملية مقارنة و استبدال بين عدة أزواج من العناصر بنفس الوقت[32]. فخلال كل مرحلة فردية:

- يقوم كل معالج فردي(الذي دليله فردي) بمقارنة عنصره مع جاره في اليمين. فإذا كان غير مرتب يتم التبادل بينهما. و في كل مرحلة زوجية:

[32] : : Daniel Etiemble, ‘ **Algorithmes de tri parallèles’**, Paris, 2003,p.3

– يقوم كل معالج زوجي (الذي دليله زوجي) بمقارنة عنصره مع جاره في اليمين . فإذا كان غير مرتب يتم التبادل بينهما. هذه الصيغة المتوازية معروضة في الخوارزمية (3–5).

الخوارزمية (3–5): الصيغة المتوازية لخوارزمية الفرز بالإبدال الزوجي – الفردي، على عدد n عملية بشكل حلقة.

```
Procedure ODD- EVEN-PAR (n)
Begin
    Id :=process's label   // process  number
    For i := 1 to n do      //n: number  of processes
    Begin
        If i is  odd then          //i%2==1 odd iteration
           If id is odd then      //odd process number
              Compare-exchange _min  (id+1) ; // Compare-exchange        to  the
right
           else
        Compare-exchange _ max (id-1) ;// Compare-exchange  to the left
  If i is even then              //i%2==0 even iteration
           If id is even then  // even process number
        Compare-exchange _min (id+1) ; // Compare-exchange      to the right
     else
        Compare-exchange _ max (id-1) ; //Compare-exchange      to the left
End for
End ODD- EVEN-PAR
```

المصدر1: **Introduction to parallel Algorithms and Parallel Program design**", University of Oregon , p.57

المصدر2: Vipin Kumar, George Karypis, Anshul Gupta, Ananth Grama, "**Introduction to parallel Computing:Sorting Algorithms** ", Addison Wesley, 2003, p.39

المصدر3: Gergel V.P، **Introduction to Parallel Programming: Parallel Methods for Sorting** ، Nizhny Novgorod، 2005 ، ص10

في الواقع و بصفة عامة يكون عدد المعالجات P في الحاسبات أقل بكثير من عدد العناصر N (p<N). و بالتالي تقسم قائمة العناصر إلى كتل متساوية بحجم N/P عنصرا. تسند كل كتلة إلى معالج P_i حيث يقوم كل معالج بترتيب عناصره محليا باستعمال إحدى خوارزميات الفرز التسلسلية على غرار الفرز السريع مثلا. فخلال كل مرحلة من الخوارزمية ، سواء منها الزوجية أو الفردية تؤدي عملية مقارنة و استبدال مع الجار الأيمن.

الخوارزمية (3-6): الصيغة المتوازية لخوارزمية الفرز الإبدال" فردي- زوجي" على p معالج (processors)(p<N):

تقسم قائمة العناصر بالتساوي بين P إجرائية (أو معالج). يكون لكل معالج (N/P) عنصرا. في البداية كل معالج يقوم بترتيب عناصره باستعمال خوارزمية فرز معينة (خوارزمية الفرز السريع مثلا). يتم الفرز على مراحل. ففي كل مرحلة فردية [33]:

1- لكل الإجرائيات الفردية وإجرائيات اليمين

- الإجرائية الفردية ترسل قائمتها مرتبة لإجرائية اليمين.
- اجرائية اليمين ترسل قائمتها مرتبة لإجرائية اليسار.
- كل اجرائية تقم بدمج قائمتها مع قائمة جارتها.
- اجرائية اليسار تحتفظ بالنصف السفلي للقائمة المدمجة و اجرائية اليمين تحتفظ بالنصف العلوي للقائمة. و في كل مرحلة زوجية:

2- لكل الاجرائيات الزوجية و اجرائيات اليمين

- الاجرائية الزوجية ترسل قائمتها مرتبة لإجرائية اليمين.
- اجرائية اليمين ترسل قائمتها مرتبة لإجرائية اليسار.
- كل اجرائية تقوم بدمج مرتب لقائمتها مع قائمة جارتها.
- اجرائية اليسار تحتفظ بالنصف السفلي للقائمة المدمجة و اجرائية اليمين تحتفظ بالنصف العلوي للقائمة.

[33] Daniel Etiemble: ، مصدر سبق ذكره ، ص3

و فيما يلي المثال (3−11) لتوضيح طريقة الفرز الفردي−الزوجي المتوازي.

مثال(3−11): الفرز الفردي−الزوجي (4 إجرائيات:Pi, i=1,4)

P1	P2	P3	P4	إجراء
13 7 12	8 5 4	6 1 3	9 2 10	
7 12 13	4 5 8	1 3 6	2 9 10	فرز محلي
7,12,13,4,5,8 4 5 7	4,5,8,7,12,13 8 12 13	1,3,6,2,9,10 1 2 3	2,9,10,13,6 6 9 10	فردي
4 5 7	8,12,13,1,2,3 1 2 3	1,2,3,8,12,13 8 12 13	6 9 10	زوجي
4,5,7,1,2,3 1 2 3	1,2,3,4,5,7 4 5 7	6,9,10,8,12,13 6 8 9	6,9,10,8,12,13 10 12 13	فردي
1 2 3	4,5,7,6,8,9 4 5 6	6,8,9,4,5,7 7 8 9	6,9,10,8,12,13 10 12 13	زوجي

المصدر: Daniel Etiemble ، مرجع سبق ذكره، ص4.

الخوارزمية (3−7): خوارزمي الفرز السريع المتسلسل (sequential quick sort algorithm).

خوارزمية الترتيب السريع هي طريقة من نوع فرق− تسد ، تقوم بتقسيم عناصر قائمة T إلى قائمتين فرعيتين T1 و T2 و اللتان تحتويان على التوالي على أكبر العناصر و أصغر العناصر بالنسبة لعنصر محوري فيها، ثم بعد ذلك يتم تطبيق الخوارزمي تراجعيا على القائمتين الفرعيتين T1 وT2 .

هناك العديد من الطرق لتقسيم القائمة. المبدأ العام يتمثل في استعمال عنصر معين يسمى المحور(Pivot) كقيمة تقسيم. يمكن اختيار العنصر الأول في القائمة كما هو مبين في المثال (3−12)

(أنظر الصفحة 110) كما يمكن اختياره عشوائيا من القائمة. للعنصر المحور أهمية في تسريع عملية الترتيب ، لذا فاختياره أساسي.

خوارزمية الفرز السريع المتسلسل هو إجراء تراجعي يعمل على النحو التالي:[34]

1. يتم تحديد أحد العناصر كمحور القائمة.

2. تقسيم القائمة إلى قائمتين فرعيتين : قائمة سفلى تحتوي على عناصر أصغر من المحور، وقائمة عليا تحتوي على عناصر أكبر أو يساوي المحور.

3. القائمة السفلى والقائمة العليا تعيد الإجراء تراجعيا لفرز أنفسهما.

4. النتيجة النهائية هي دمج القائمة السفلى مرتبة، المحور، والقائمة العليا مرتبة كذلك.

و فيما يلي نقدم أحد الخوارزميات و الذي تتجسد من خلاله طريقة الفرز السريع المتسلسل.

```
// sequential quick sort algorithm
quicksort(list, left, right)
{
   if(left < right)
         {
         q=partition(list, left, right) ;  //q:indice pivot
         quicksort(list, left, q-1);      // left list
         quicksort(list, q+1, right);  // right list
         }
}
// partition of list
partition(list, left, right)
{
```

[34] Thomas Cormen, Charles Leiserson, Ronald Rivest, Clifford Stein, "I**NTRODUCTION À L'ALGORITHMIQUE",** Dunod, Paris, 2004, pp.139-140

```
 p=left;
 pivot= list[left];
 for (i=left+1; i<=right, i++)
 {
   if(pivot > list[i])
  {
    list[p] = list[i];
    list[i] = list[p+1];
    list[p+1] = pivot;
    p= p+1;
 }
}
 return p;}
```

المصدر: Maha Saada &Huda Saada&Mohammad Qatawneh, "**Performance Evaluation of Parallel Sorting Algorithms**", International of Advanced Science and Technology, Vol.95(2016), p.3

الخوارزمية (3−9):الصيغة المتوازية لخوارزمي الفرز السريع (عنوان مشترك)

يعتمد في هذا الخوارزمي على الخطوات التالية[35]:

1. parallelquicksort(A, q, r)

// رتب قائمة الأعداد A[q..r] على عدد المعالجات.

2. البداية

3. إنشاء عدد من المعالجات P

[35] : Zaineb T. Baqer, ' **Parallel computing for sorting Algorithms**', Baghdad science journal, vol. 11(2), 2014, p.8

// الصياغة هي من نوع عنوان مشترك (shared address type)

4. تقسيم القائمة A إلى كتل من n/p

5. تعين الكتلة Ai من القائمة إلى المعالج Pi.

6. يحدد المعالج الرئيسي العنصر المحور(pivot).

7. يرسل المعالج الرئيسي العنصر المحور إلى جميع المعالجات الأخرى.

8. إيعاد ترتيب (rearrange(Ai, Si, pivot

// يقسم كل معالج كتلته إلى كتلتين فرعيتين Si بعناصر أصغر من عنصر المحور و Li بعناصر أكبر من عنصر المحور.

9. تخزن الكتلة Si في بداية القائمة A.

10. يقوم المعالج الرئيسي بتقسيم المعالجات إلى قسمين:

11. إذا كان المعالج في المجموعة الأولى (parallelquicksort(S, left of S, right of S

12. إذا كان المعالج في المجموعة الثانية (parallelquicksort(L, left of L, right of L

13. نهاية parallelquicksort

الخوارزمية (3-10): خوارزمي الفرز السريع على مكعب(Parallel quicksort on hyper cube)

يعتمد في هذا الخوارزمي على الخطوات التالية [36]:

1- تقسيم القائمة غير المرتبة على كل عقدة (معالج).

2- ترتيب كل عقدة لبينتاها محليا.

3- انطلاقا من العقدة 0 ، يتم توزيع القيمة الوسيطة.

4- تقسيم كل قائمة محليا، ثم استبدال كل النصفين على أعلى بعد.

5- إعادة الخطوات 3و 4 حتى بلوغ البعد 0 .

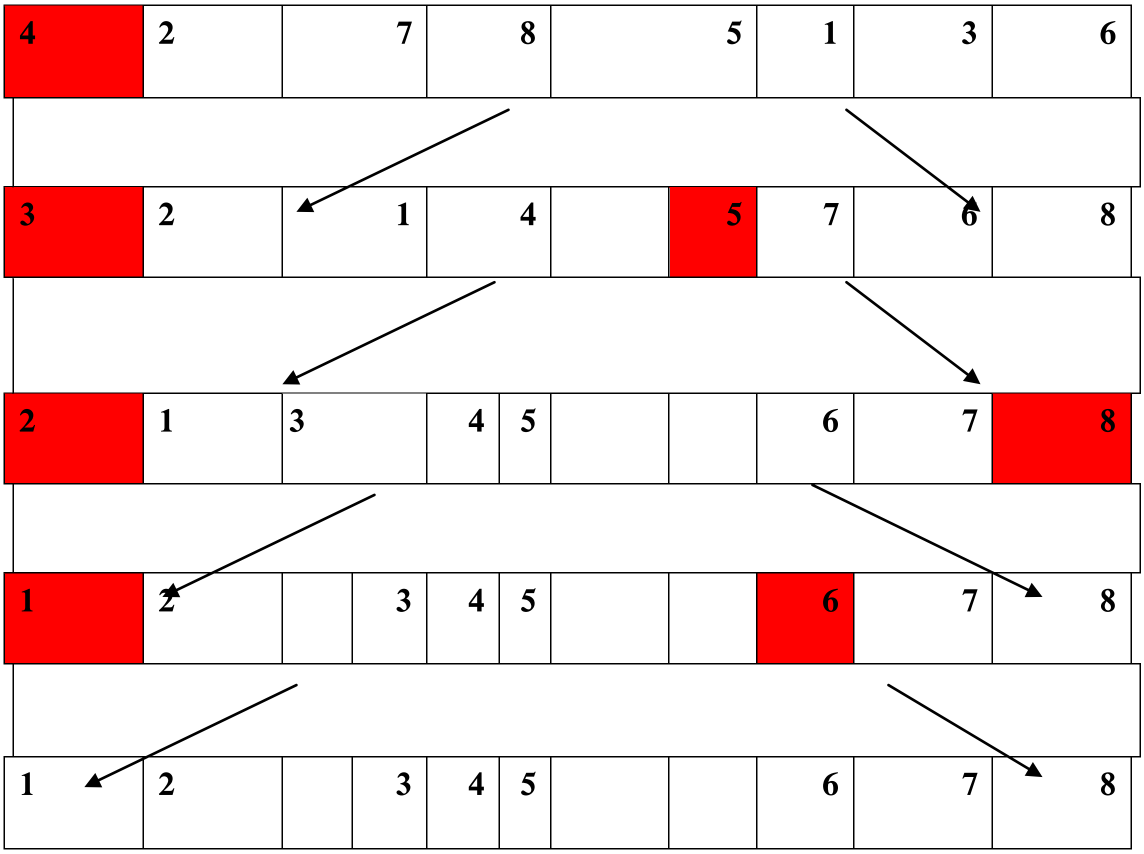
[36] **Parallel Algorithm Quick Guide**: ، مصدر سبق ذكره ،ص23

```
procedure ParallelQuickSortHpercube(B, n)
//sort  sequence  B of  size n  on d dimensional  hypercube. p=2^d is number of processes.
begin
id:= process's label;
  for i:= 1 to d do
    {
      x:= pivot;
      partition B  into  B1 and B2 such  that  B1<=  X < B2;
     if  i^th bit is 0 then
    {
              send  B2  to  the  process  along the i^th communication link;
               C:= subsequence received along the i^th communication link;
               B:= B1 U C;
       }
        else
         {
               send  B1  to the process along  the  i^th  communication   link;
                C:= subsequence received  along the i^th  communication  link;
                B:= B2 U C;
       }
 }
  sort B using sequentiel quick sort;
end Parallel QuickSortHperCube
```

المصدر1: Zaineb T. Baqer, ‘ **Parallel computing for sorting Algorithms**’, Baghdad science journal, vol. 11(2), 2014, p.9

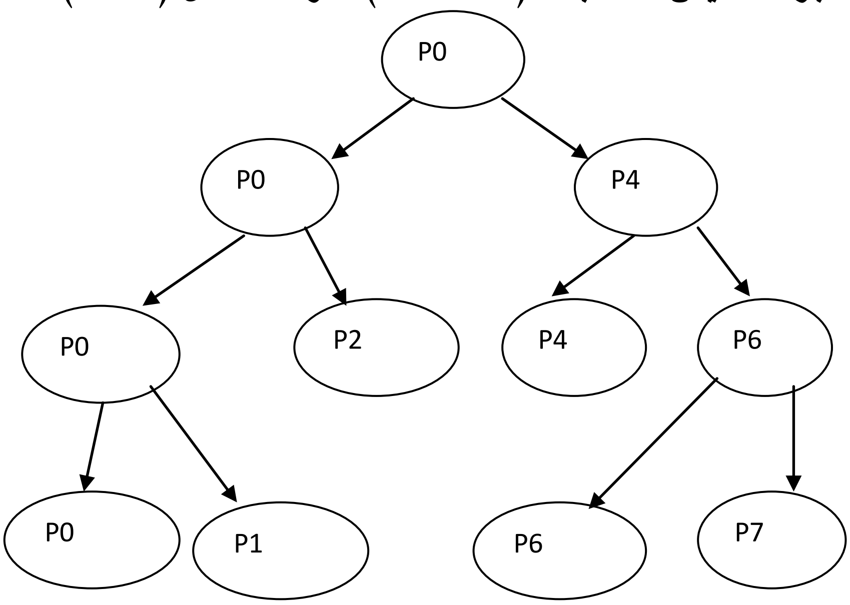
المصدر2: **Parallel Algorithm Quick Guide** , P.23

مثال (3-12): الخوارزمي المتوازي الفرز السريع باستعمال شجرة التخصيص للمعالجات.



المصدر : ‘ Fernando Silva,’ Parallel Algorithms sorting ، ص15

الشكل (3-14): شجرة تخصيص المعالجات (Pi i=1,4) الموافقة للمثال (3-12)



المصدر : ‘ Fernando Silva,’ Parallel Algorithms sorting ، ص15

الخوارزمية(3−11):خوارزمي الفرز بالدمج (Merge Sort):

خوارزمية الفرز بالدمج هي طريقة من نوع فرق−تسد . تقسم القائمة محل الفرز إلى قائمتين فرعيتين بطول متساوي ، ثم بعد ذلك ترتب كل واحدة منها على حدا تراجعيا و في الأخير يتم دمجها معا في قائمة بحيث تكون هي كذلك مرتبة. فمبدأ الطريقة هو مقارنة القائمتين عنصر بعنصر ثم وضع العنصر الأصغر/الأكبر في مكانه الصحيح على التوالي في القائمة الأم حسب وهذا حسب نوع الفرز تصاعدي أو تنازلي، و هكذا حتى نهاية أخذ جميع عناصر القائمتين الفرعيتين. ينهي التراجع عندما يصبح طول القائمة الفرعية واحد. لأن هذه الأخير تكون دائما مرتبة. العملية الأساسية في الفرز بالدمج هي طبعا قائمتين مرتبتين.[37]

و فيما يلي نص خوارزمي الفرز بالدمج المتوازي لتجسيد طريقة فرز القائمة A.

الخوارزمية (3−11): خوارزمي الفرز بالدمج المتوازي(Parallel Merge Sort)

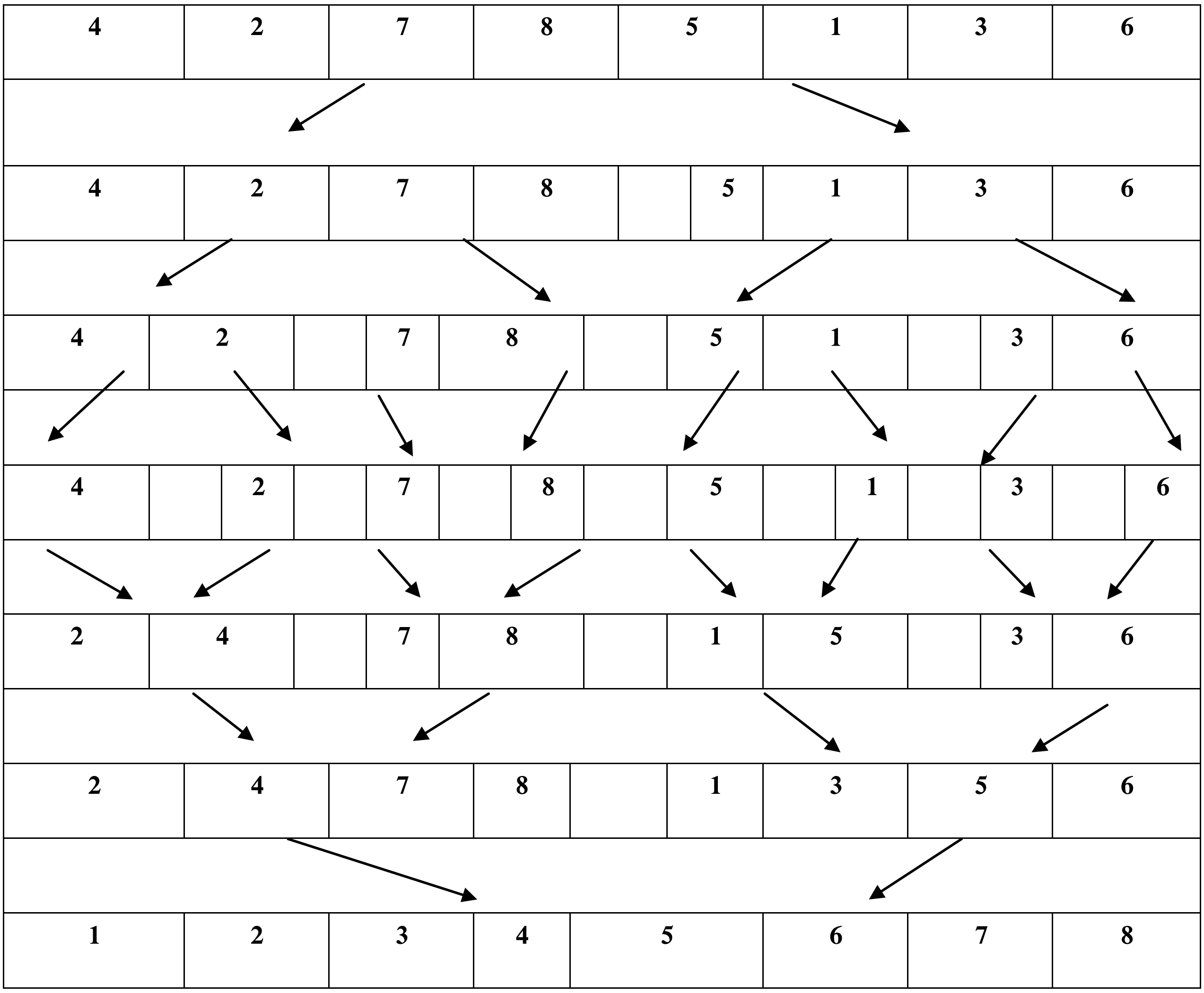
```
Algorithm :MergeSort( A )  // A: list of elements
1.  if (|A| = 1)  then return A
2.      else
3.            in parallel  do
4.            L:= MergeSort(A[0: |A|/2]);     // left list
5.            R:= MergeSort ([|A|/2 : |A|])   // right list
6.            return Merge(L,R )
```

المرجع: Blellouch ,Guy E.، Parallel Algorithms Sorting ، ص.19

[37]: Fréderic Vivien ، مصد سبق ذكره ، صص.29-30

نقوم في المثال(3−13) بتوضيح طريقة عمل الفرز بالدمج لخوارزمي(3−11).

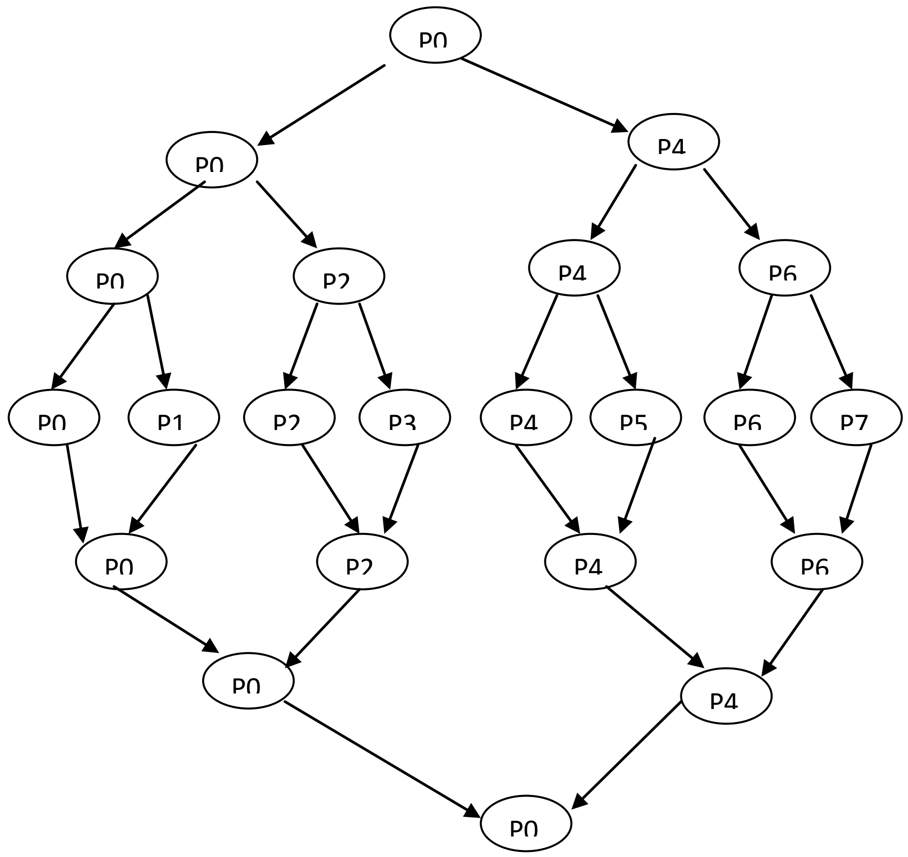
مثال (3−13) : الخوارزمي المتوازي للفرز بالدمج باستعمال شجرة التخصيص للمعالجات لقائمة الأعداد(4, 2, 7, 8, 5, 1, 3, 6).



المصدر : ' Fernando Silva,' Parallel Algorithms sorting ، ص 14

في البداية يتضح من خلال المثال(3−13) أن القائمة كانت غير مرتبة في الصف الأول من الجدول ، ثم بعد ذلك يتم تقسيمها تراجعيا إلى قوائم فرعية حتى تصبح كل قائمة فرعية مكونة من عنصر واحد فقط. بعد ذلك يتم ترتيبهما و دمجهما مها. في النهاية نحصل على القائمة مرتبة كما هو واضح في نهاية الجدول.

الشكل (3−15): شجرة التخصيص للمعالجات الموافقة للمثال(3−13) خوارزمي الفرز بالدمج.



المصدر : Fernando Silva ، مصدر سبق ذكره ص 14

يوضح كل من الشكل(3−14) و الشكل(3−15) خريطة أو كيفية الاسناد أو التعيين الأمثل للمعطيات في القائمة بين مختلف المعالجات.

في الشكل(3−15) ،أسندت القائمة في البداية إلى المعالج الجذر P0، ثم تقسم إلى قائمتين متساويتين و تسند اليسرى إلى المعالج P0 و اليمنى إلى المعالج P4 و هكذا تراجعيا حتى يصبح لكل معالج عنصر واحد فقط . بعد ذلك تدمج كل زوج من هذه القوائم الفرعية مع ترتيبها حتى نحصل على القائمة النهائية مرتبة و مسندة للمعالج P0 .

خلاصة :

حاولنا في هذا الفصل توضيح طريقة تصميم الخوارزميات المتوازية. و قلنا أنه هناك طريقتان أساسيتان لبناء خوارزمي متوازي. الأولى تتمثل في تحديد و استغلال التوازي الموجود ضمنيا في الخوارزمي التسلسلي و الثانية تتمثل في إنشاء خوارزمي متوازي جديد. و يمكن أن تتبع طريقة التصميم الخطوات التالية:

- ✓ تحديد أجزاء العمل(المسألة) التي يمكن أن تؤدى بشكل متزامن.
- ✓ إسناد الأجزاء المتزامنة من العمل إلى عدة معالجات تعمل بالتوازي.
- ✓ توزيع المعطيات المدخلة و المخرجة و الوسيطة المرتبطة بالخوارزمي.
- ✓ إدارة عملية الوصول للمعطيات المشتركة بين هذه الإجرائيات أو المعالجات. و يعتمد في هذه الطريقة على إعداد مخططات بيانية مثل بيان التبعية و الذي عقده تمثل المهام(معطيات) و الأضلاع ، التبعية بين المهام (أو التواصل بين المهام) و كذلك مخططات لعمليات اسنادها للمعالجات. و لتوضيح ذلك أعطينا مثال أول لعملية ضرب مصفوفة بشعاع و مثال ثاني لعملية في قواعد البيانات.

تطرقنا كذلك إلى مختلف تقنيات التقسيم و ذكرنا مثلا التقسيم البيانات ، التقسيم العودي أو التراجعي ، التقسيم الاستكشافي و التقسيم التخميني و وضحنا كل منها بأمثلة و الخوارزميات على غرار خوارزمي الترتيب الفردي–الزوجي و خوارزمي الفرز السريع في صيغها التسلسلية و المتوازية و أمثلة توضيحية.

الفصل الرابع

استخدام الخوارزميات االمتوازية في حل مسائل نظرية البيان

تمهيد:

لقد تطور استعمال علم بحوث العمليات تطورا ملحوظا خاصة في ظل تزامنه مع التطور العلمي الكبير الذي تم تحقيقه في مجال الحاسبات الآلية . كما يمكن أن نلخص أهمية بحوث العمليات فيما يلي :

- تعتبر وسيلة مساعدة في اتخاذ القرارات الكمية باستخدام الطرق العلمية الحديثة .
- يعتبر علم بحوث العمليات من الوسائل العلمية المساعدة في اتخاذ القرارات بأسلوب أكثر دقة وبعيد عن العشوائية الناتجة عن التجربة والخطأ .
- تعتبر بحوث العمليات فن وعلم في آن واحد فهي تتعلق بالتخصيص الكفء للموارد المتاحة وكذلك قابليتها الجديدة في عكس مفهوم الكفاءة والندرة في نماذج رياضية تطبيقية .
- يسعى هذا العلم إلى البحث عن القواعد والأسس الجديدة للعمل الإداري ، وذلك للوصول إلى أفضل المستويات من حيث الجودة الشاملة ، ومقاييس المواصفات العالمية .
- أنها تساعد على تناول مشاكل معقدة بالتحليل والحل والتي يصعب تناولها في صورتها العادية .
- أنها تساعد على توفير تكلفة حل المشاكل المختلفة وذلك بتخفيض الوقت اللازم للحل .
- أنها تساعد على تركيز الاهتمام على الخصائص الهامة للمشكلة دون الخوض في تفاصيل الخصائص التي لا تؤثر على القرار ، ويساعد هذا في تحديد العناصر الملائمة للقرار واستخدامها للوصول إلى الأفضل .

تعتبر بحوث العمليات و الأساليب الكمية بصفة عامة، أسلوب رياضي يتم من خلاله معالجة مختلف المشاكل الاقتصادية والإدارية، وذلك بمساندة الموارد المتاحة من بيانات وأدوات والطرق التي تستخدم من قبل متخذي القرار لمعالجة المشاكل. هذا من جهة ومن جهة أخرى، الترشيد هو البحث عن حالة العقلانية لأي تصرف أو سلوك إنساني، ويقصد بترشيد القرارات إضفاء صفة العقلانية على القرار المتخذ بحيث يتحقق الاستخدام الأمثل والصحيح لكل الإمكانات المتاحة.

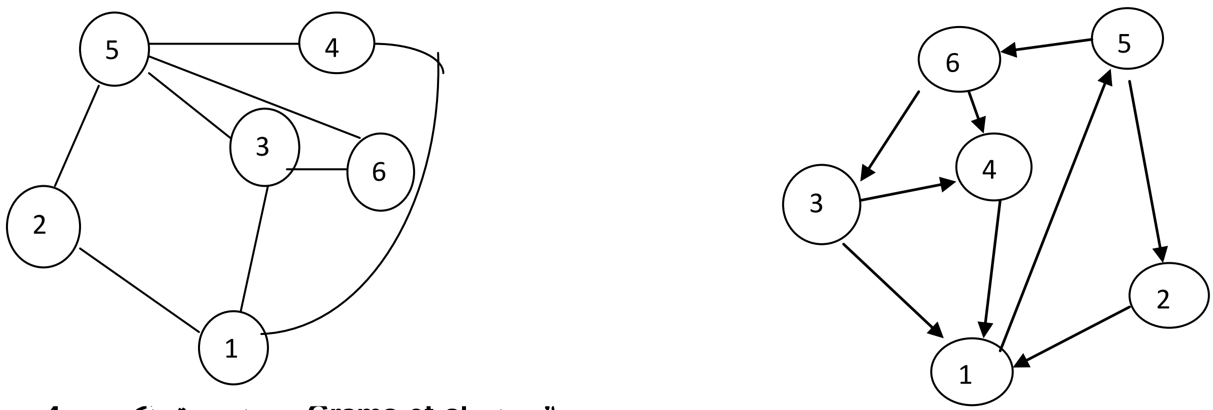
إن مبدأ الترشيد لأي عملية اتخاذ القرار يجب أن يتم على أساس علمي مدروس حيث أن العشوائية والحدس في اتخاذ القرار تعتبر غير مقبولة بشكل عام، إضافة إلى أنها لم تعد مناسبة بشكل قاطع بسبب التطورات الاقتصادية والتكنولوجية السريعة التي حدثت وما ترتب عن ذلك من تعقيد وصعوبات اتخاذ القرارات. ولهذا لابد من استخدام منهج علمي يقوم على الأساليب الكمية لترشيد عملية اتخاذ القرارات.

تلعب نظرية البيان (Graph Theory) و هي إحدى فروع بحوث العمليات دوراً هاماً في علم الحاسب الآلي لأنها توفر طريقة سهلة و منهجية لنمذجة العديد من المسائل. و يمكن التعبير عنها من خلال **البيان (Graph)** كما يمكن حلها باستخدام خوارزميات بيانية قياسية.

4 .1 تعريف و مفاهيم أساسية: نقوم في ما يلي بإعطاء تعريفات لبعض المفاهيم في نظرية البيان نراها ضرورية في هذه الدراسة[1].

4 .1 .1 البيان (Graph) : هو الثنائية G=(V,E) و يتألف البيان G من مجموعة من القمم V ، و مجموعة من الأضلاع E ، بحيث أن كل ضلع يصل بين قمتين من القمم. يوجد نوعان من البيان : بيان موجه ، و بيان غير موجه ، في البيان الموجه يكون لكل ضلع اتجاه واحد فقط ، في حين أن البيان الغير موجه يكون للضلع اتجاهين ، فمثلاً و على افتراض أن القمتين u و v ينتميان إلى مجموعة القمم V ، وكان هنالك ضلع e يصل بين القمتين (u,v) ففي البيان غير الموجه نقول أن القمتين u و v متصلان. أما في البيان الموجهة فنقول أن هناك اتصال من u إلى v [2]. و هو ما يوضحه الشكل(4−1)

الشكل (4−1): (a) بيان موجه ، (b) بيان غير موجه.



المصدر:Grama et al، مصدر سبق ذكره، ص4

إذا كان (u,v) ضلع في بيان غير موجه G=(V,E) فإنه يقال أن القمتين u و v أنهما مجاوران لبعضهما البعض. أما في حال كان الضلع في بيان موجه فإننا نقول أن القمة v مجاور القمة u و ليس العكس.

[1] : Cyril Gavoile, "**Algorithmes Distribués**, Université de Bordeaux, 2016, p20

[2] : Loic Helouet, '**Algorithmes et graphes**', pp1-5

و من أهم العناصر التي يجب تحديدها في دراسة البيان ما يلي[3]:

4 .1 .2 درجة القمة u (deg(u)): هو عدد القمم المجاورة للقمة u . أي عدد القمم الموالية له. و لدينا العبارة $\sum_{u \in V} \deg(u) = 2m$.

أين m : هو عدد الأضلاع في البيان(كل ضلع يساهم مرتين في درجة قمتيه).

المسار: إن المسار من القمة v إلى القمة u هو تتابع $<v_0,v_1,v_2,\ldots,v_k>$ من القمم بحيث أن $v_0=v$ و $v_k=u$ ، و أن (v_i,v_{i+1}) تنتمي إلى E من أجل $i=0,1,..,k-1$. المسار يصبح دارة عندما $v_0 = v_k$.

4 .1 .3 طول المسار(dist(u,v)): هو عدد الأضلاع الموجودة في المسار (أي التي تكوّن المسار).

أقصر مسار: أقصر مسار بين القمة u و القمة v هو المسار ذي الطول الأدنى[4].

4 .1 .4 البيان المتصل (connex graph): يكون البيان G متصلا إذا وجد مسارا بين كل زوجين من القمم. إذا كان البيان متصلا تكون العلاقة كالتالي: $m > n-1$ (m: عدد الأضلاع ، n: عدد القمم).

4 .1 .5 وزن الضلع : يقرن في بعض الأحيان وزن لكل ضلع من E . و الوزن w في الغالب عدد حقيقي يمثل كلفة أو منفعة العبور للضلع. و البيان الذي له أوزان ترتبط مع كل ضلع يدعى بأنه بيان موزون، و يمكن أن يشار إليه G=(V,E,W)، حيثVهي القمم وE هي الأضلاع كما أشرنا سابقا ، أما $W: E \rightarrow R$ فهي تابع حقيقي معرف علىE نحو R.

4 .1 .6 وزن البيان: هو مجموع أوزان أضلاعه.

4 .1 .7 وزن المسار:هو مجموع أوزان الأضلاع المكونة له.

4 .1 .8 قطر البيان: قطر البيان G هي القيمة المشار لها بالعلاقة الرياضية التالية:

$$Diam(G)= \max_{u,v \in V} dist(u,v)$$

و هي أكبر مسافة في البيان.

3: Cyril Gavoile ، مصدر سبق ذكره ، ص20

4: Loic Helouet ، مصدر سبق ذكره ، ص5

4 .1 . 9 الانحراف **(Eccentricity)** :ecc(u) هي المسافة التي تفصل u عن أبعد قمة في البيان G. يشار لها:

$Ecc(u)= \max_{v\epsilon V} dist(u,v)$

و هو أيضا العلو الأدنى في شجرة التغطية و الجذر u.

نشير كذلك أن:

$Diam(G) = \max u_{\epsilon V} ecc(u)$

و تسمى هذه القيمة (ecc(G الانحراف من الرسم البياني(eccentrity of the graph).

4 .1 .10 درجة البيان ((deg(G)): درجة البيان هي أقصى درجة لقممه.

4 .1 .11 المسار الهاملتوني(Hamiltonian path): هو المسار الذي يمر عبر جميع قمم البيان مرة و مرة واحدة فقط.

4 .2 طرق تمثيل المخططات البيانية في الحاسوب.

هناك طريقتان قياسيتان لتمثيل المخططات البيانية في برامج الحاسب .

الأولى باستخدام المصفوفات ، و الثانية باستخدام القوائم المتصلة Linked List .

4 .2 .1 طريقة المصفوفات لتمثيل البيان:

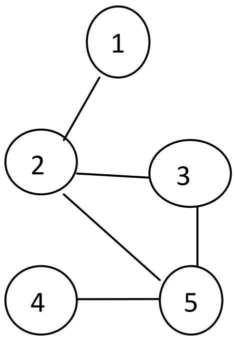
ليكن لدينا البيان(G=(E,V و فيه القمم مرقمة من 1 و حتىn . مصفوفة الجوارadjacency matrix لهذا البيان هي المصفوفة($A= (a_{i,j}$ و لها الحجم n×n، و معرفة كالتالي[5]:

[5]: Grama et. al. , ‘**Parallel Graph Algorithm**’, 1994, pp.6-7

في الشكل (4−2) توضيح لبيان غير موجه ممثل بمصفوفة جوار. و يلاحظ أن مصفوفة الجوار للبيان الغير موجه هي مصفوفة متناظرة . و يمكن أن يعدَل التمثيل في مصفوفة الجوار لتتماشى مع البيانات الموزونة(weighted graphs) . و في هذه الحالة يمكن تعريف($a_{i,j}$) =A على النحو الآتي:

الشكل (4−2): بيان غير موجه و تمثيله بمصفوفة الجوار.

$$a_{ij} = \begin{cases} w(v_i, v_j) if (v_i, v_j) \in E \\ 0 if i = j \\ \infty otherwise \end{cases}$$



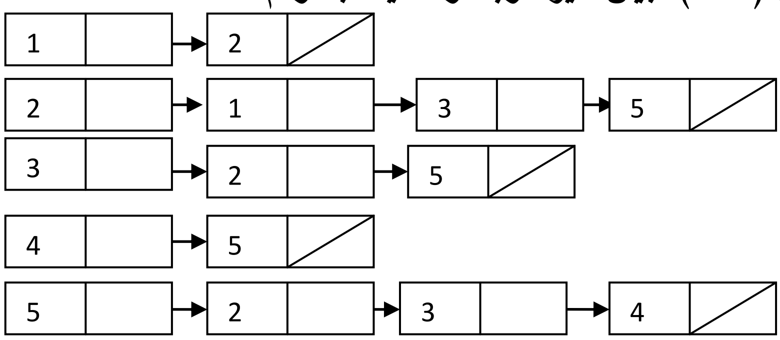
$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

المصدر:.Grama et. al، مصدر سبق ذكره، صص5−6

سنشير لمصفوفة الجوار المعدلة بمصفوفة جوار الموزونة .

4 .2 . 2 طريقة القوائم المتصلة لتمثيل البيان: سنقوم بتمثيل البيان السابق باستعمال طريقة القوائم المتصلة حسب الشكل التالي

الشكل (4−3): بيان غير موجه و تمثيله بالقوائم المتصلة.



المصدر:.Grama et. al، مصدر سبق ذكره، ص7

4 .3 أمثلة لخوارزميات في بحوث العمليات: نقوم فيما يلي بإعطاء أمثلة لبعض الخوارزميات المعروفة في نظرية البيان كخوارزمية Prim و خوارزمية Dijkstra لإيجاد الشجرة بأقل تغطية في بيان غير موجه

و كذلك خوارزميات كل من Floyd و Kruskal لإيجاد أقصر مسار. و هذا نظرا لأهمية استعمالاتهم في ميدان حل بعض المسائل الاقتصادية.

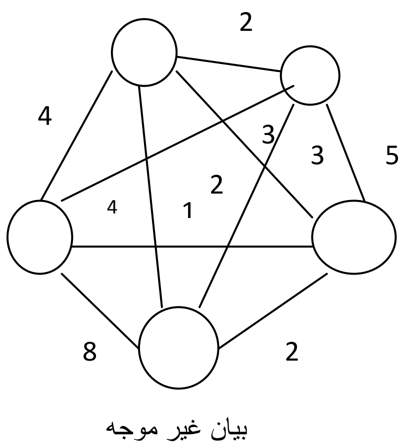
4 .3 .1 الشجرة المثلى :(خوارزمية Optimal Tree:Prim's Algorithm (Prim

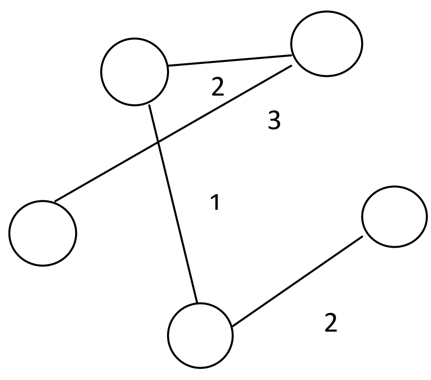
إن الشجرة الأمثل لبيان غير موجه G هي بيان جزئي من G يكون شجرة على جميع قمم G. وفي البيان الموزون ، يكون الوزن للبيان الجزئي هو مجموعة أوزان الأضلاع فيه، و الشجرة التغطية الأمثل لبيان موزون غير موجه هي شجرة لها الوزن الأمثل(أصغر/أكبر) . و تتطلب العديد من المسائل إيجاد شجرة بأقل تغطية لبيان غير موجه . ويمكن تطبيق الشجرة بأقل تغطية في المجالات الاقتصادية التالية :

- إيجاد الطول الأقصر للكابلات التي تربط مجموعة حاسبات في شبكة.
- تحديد أقل كلفة لربط خطوط الاتصالات بين المدن.
- تحديد أقل كلفة لربط شبكة توزيع المياه بين السكنات.
- تحديد أقل كلفة لربط شبكة توصيل الكهرباء.

يمكن أن يتم عن طريق البحث عن شجرة أقل تغطية للبيان غير موجه الذي يحتوي على كل الارتباطات كما يوضحه الشكل(4-4)

الشكل (4-4): بيان غير موجه ، و شجرة بأقل التغطية فيه.





المصدر:George Karypis,"Introduction to parallel computing", p4

إذا لم يكن G متصلاً، فإنه لا يمكن أن يكون له شجرة تغطية . بل سيكون لديه غابة تغطية spanning forest، و من أجل تبسيط فكرة حساب شجرة بأقل تغطية نفرض أن G متصل. فبصفة

عامة تبدأ الطريقة باختيار عشوائي لقمة البداية والتي تسمى جذر الشجرة ، بعد ذلك نشرع في تكوين الشجرة انطلاقا من الجذر على مراحل بحيث تكون الشجرة ذات تغطية أو وزن أمثل و هذا باختيار و إضافة في كل مرحلة قمة (ضلع) من بين القمم المتبقية والتي تحقق شرط التغطية الأمثل (أقل/أكبر) . تتوقف الطريقة عندما تكون جميع القمم موجودة في الشجرة [6].

أ- خوارزمية Prim التسلسلية لإيجاد شجرة بأقل تغطية[7].

```
1.  procedure PRIM_MST(V, E, w, r)
2.   begin          //initialize vertices with edge to r(root vertex) is given
3.      V_T := {r};   // cost = weight of the edge. all other vertices the cost=∞
4.      d[r] := 0;                  // compute d[:], the weight between r and
5.      for all v ϵ (V − V_T ) do   // and each vertex outside V_T
6.         if edge (r, v) ∃  set d[v] := w(r, v);
7.         else set d[v] :=∞ ;
8.      while V_T ≠ V do          // while there are vertices outside T
9.      begin
                                  // use d[:] to find u vertex closest to T
10.        find a vertex u such that d[u] = min{d[v]|v ϵ (V − V_T )};
11.         V_T := V_T U {u};              // add u to T
12.         for all v ϵ (V − V_T ) do       // recompute d[:] now
13.             d[v] := min{d[v], w(u, v)};
14.      endwhile
15.  end PRIM_MST
```

Grama et. al. , 'Parallel Graph Algorithm', 1994,p.10:1المرجع

Vivec Sarkar , "Parallel Graph Algorithms", 2008, p.13:2 المرجع

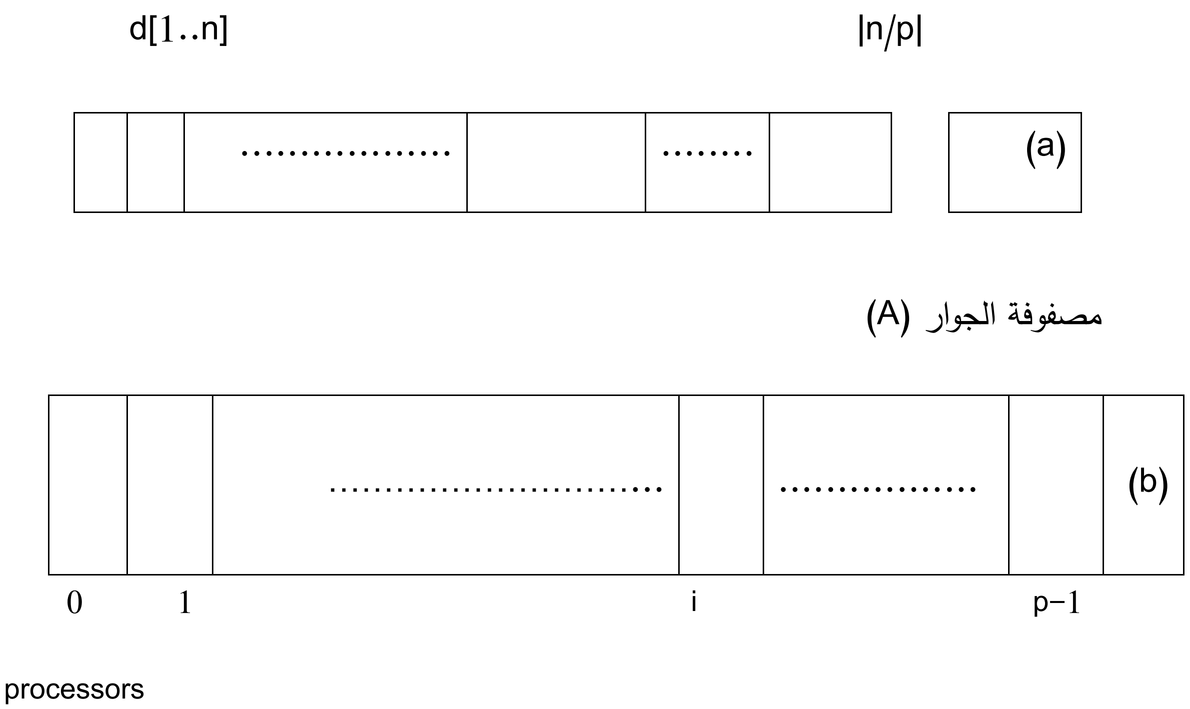
ب- الصيغة المتوازية لخوارزمية Prim : نلاحظ من خلال نص خوارزمي Prim أنه يتم في كل مرحلة أو تكرار اخيار و إضافة قمة واحدة لمجموعة قمم شجرة التغطية (V_T) و هذا ما يظهر في التعليمة (السطر 10) من الخوارزمي و بالتالي لا يمكن إضافة قمتين في نفس الوقت. بمعنى يمكن

[6] : Vivec Sarkar , "Parallel Graph Algorithms", 2008, pp.8-9

[7]:" Introduction to parallel computing", University of Oregon, IPCC, p.68

التعليمة while (السطر 8) و بالمقابل يمكن ايجاد التوازي الضمني في الحلقة الداخلية for . و قصد تحقيق هذا الهدف نتبع الطريقة التالية: في البداية تقسم مصفوفة الجوار A للبيان G إلى كتل أعمدة بحجم (n/p) بالتساوي ، و نفس الشيء نفعله مع شعاع حساب المسافات d[:] . الشكل(4-5) يوضح هذا التقسيم. ثم نسند كل كتلة B_i لمعالج P_i ($i= 0,1,2,....p-1$) و جزء شعاع المسافات d_i[:] الموافق لمجموعة قمم V_i ، ليقوم بعد ذلك و خلال كل تكرار كل معالج P_i بمعالجة شعاع المسافات d_i[:] . لنحصل على أصغر قيمة في كل معالج ، بعد ذلك يتم حساب أدنى قيمة شاملة d[u] و تخزن في المعالج الجذر P_0 حسب نوع التواصل المختصر الكل مع واحد (all to one reduction)، يقوم في هذه الأثناء المعالج الجذرP_0 بتوزيع القمة المختارة u لبقية المعالجات الأخرى و بإضافة القمة u لمجموعة القمم V_T. وأخيرا ، يقوم كل معالج بتحديث قيم d[v] للقمم الخاصة به (المحلية) يتوقف الخوارزمي عندما تصبح مجموعة قمم البيان هي قمم شجرة التغطية[8].

الشكل (4- 5) تقسيم المصفوفة d ومصفوفة الجوار A الى P اجرائية(أو معالج)



المصدر1:.Grama et. al، مصدر سبق ذكره ، ص13 ،المصدر2: , Introduction to parallel computing University Oregon.، مصدر سبق ذكره ، ص72

[8] Grama et. al.: ، مصدرسبق ذكره ، ص12

عند اضافة قمة جديدة u الى V_T، فان قيم d[v] التي تحقق v ينتمي إ لى($V-V_T$) يجب أن يتم تحديثها. الإجرائية المطابقة ل v يجب أن تعلم وزن الضلع (u,v) و بالتالي ستحتاج كل إجرائية P_i إلى تخزين عمود مصفوفة الجوار الموزونة المطابق للمجموعة V_i للقمم المسندة إليها.

4 .3 .2 أقصر مسار:

يوجد العديد من المسائل الاقتصادية التي يمكن نمذجتها من خلال نظرية البيان نذكر على سبيل المثال:

– مسائل المسار الأمثل(اختيار مسلك في شبكة طرقات).

– مسائل الدارات (مشاكل تشبع الشبكات).

– المسار الهاملتوني(مسألة التاجر المتجول الذي يجب أن يعود لمدينته بعد زيارته لزبائنه في كل مدينة زارها).

يستعمل خوارزمي**DIJKSTRA** لإيجاد أقصر مسار بين قمة بداية باتجاه كل قمة أخرى في البيان G [9].

أ– خوارزمي DIJKSTRA التسلسلي

```
procedure DIJKSTRA_SP(V, E, w, s)
 begin
    V_T := {s};
    for all v ∈(V − V_T ) do
       if edge (r, v) ∃ set l[v] := w(s, v);
       else set l[v] :=∞ ;
    while V_T ∈ V do
    begin
       find a vertex u such that l[u] := min{l[v]|v∈  (V − V_T )};
        V_T := V_T U {u};
        for all v∈  (V − V_T ) do
```

9 : "Introduction to parallel computing", University of Oregon, p74

13. l[v] := min{l[v], l[u]+w(u, v)};

14. endwhile

15. end DIJKSTRA

George Karypis ,' Introduction to parallel computing : graph Algorithms' pp.8-10 :المرجع

ب-الصيغة المتوازية لخوا رزمي DIJKSTRA: التحليل الذي قيل في عملية التوازي في خوارزمي PRIM ينطبق على خوارزمي DIJKSTRA مع تخزين l[u] والذي يمثل مجموع الأوزان أو القيم للمسار من قمة بداية المسار s حتى القمة u (أي الوزن الأدنى من القمة s حتى القمة u).

4 .3 .3 أقصر مسار بين كل زوج من القمم:

توجد العديد من الخوارزميات التي تقوم بهذا العمل منها خوارزمي فلايد و الذي يسمح بحساب مسافة أقصر المسارات بين كل زوج من القمم في البيان G=(V, E) . يفترض في هذا البيان خلوه من الدارات. تحسب مسافة المسار بين القمة i و القمة j مباشرة أو عبر قمة وسيطة k (أي من القمة i نحو القمة k و من القمة k نحو القمة j). و هو ما يقوم به الخوارزمي فلايد التسلسلي (أ) الموالي في سطره السابع .

أ-خوارزمي Floyd التسلسلي (Sequential Floyd's all pairs shorted path algorithm)

1. procedure FloydAllPairsSP(A)

2. begin

3. $D^{(0)}$:= A;

4. for k;= 1 to n do

5. for i:= 1 to n do

6. for j:= 1 to n do

7. $d_{i,j}^{(k)}$:= min ($d_{i,j}^{(k-1)}$, $d_{i,k}^{(k-1)}$ + $d_{k,j}^{(k-1)}$);

8. endFloydAllPairSP

Vivek Sarkar, "Parallel Graph Algorithms", Rice University, 2008:المرجع

ب- خوارزمي Floyd المتوازي لأقصر مسار بين كل زوج من القمم

1. **procedure FloydAllPairsParallel(A)**

2. **begin**

3. **$D^{(0)}$:= A;**

4. **for k;= 1 to n do**

5. **for all $P_{i,j}$ where i, j ≤ n do in parallel**

6. **$d_{i,j}^{(k)} := \min(d_{i,j}^{(k-1)}, d_{i,k}^{(k-1)} + d_{k,j}^{(k-1)})$;**

7. **endFloydAllParallel**

المرجع: George Karypis , ‘ Introduction to parallel computing : graph Algorithms’, pp.11-14

4 .4 تحليل الخوارزميات المتوازية :

لقد ازدادت سرعة الحواسيب كثيرا في الفترة الأخيرة, و كان يعتقد أن فعالية الخوارزميات ليست ذات أهمية كبيرة و لكن الحقيقة التي ظهرت اليوم أن الفعالية أمر مهم, و هذا ما يدعونا إلى التعمق في تحليل الخوارزميات المتوازية لمعرفة فعالياتها و أهم أسباب فعاليتها هو أن الزمن الذي تأخذه معظم الخوارزميات للتنفيذ هو دالة غير خطية في حجم إدخالها و هذا يمكنه أن ينتج بشكل أكبر قدرتها على الإفادة في زيادة السرعة ، و ليس الاعتماد فقط على سرعة الحاسبات. يحدد تحليل الخوارزميات المتوازية درجة جودة الخوارزمية و هذا يعني سرعتها و كلفة تنفيذها و مدى فعالياتها عند استعمالها في الوسائل المتاحة ، ولذلك يكون اهتمامنا بالمعايير التالية[10] : زمن التنفيذ و عدد المعالجات المستخدمة و من ثم نقوم بحساب الكلفة.

4 .4 .1 زمن التنفيذ (runtime)

إذا كانت سرعة الحاسبات هي السبب الرئيسي الذي جعلنا نهتم ببناء الحواسيب المتوازية ، فإن أهم مقياس في تقييم الخوارزمية المتوازية هو زمن تنفيذها. و يمكن تعريفه بأنه الزمن الذي تأخذه الخوارزمية خلال حل المسألة على حاسب متوازي ، يعني الزمن المستهلك من طرف الخوارزمية من بداية حتى نهاية تنفيذها. إذا كانت المعالجات المتعددة لا تبدأ و تنتهي جميعها من حساباتها بنفس الوقت عندها فإن

---

[10]: كندة زين العابدين، " **خوارزميات المعالجة المتوازية و برمجتها**"، جامعة دمشق،2006 ، ص42، ص46

زمن التنفيذ يساوي الزمن المستهلك بين اللحظة التي يبدأ فيها المعالج الأول بالحساب و اللحظة التي ينتهي فيها المعالج الأخير من الحساب. في الواقع قبل تنفيذ الخوارزمية(سواء التسلسلية أو المتوازية) على حاسب ما فإنه من المعتاد توجيه التحليل النظري للزمن الذي يتطلبه لحل المسألة الحسابية بدلالة حجم المعطيات المدخلة (n)، و عادة يكون هذا عن طريق عد عدد العمليات الأساسية أو الخطوات المنفذة من الخوارزمية في أسوأ الأحوال(ما يسمى التعقيد الزمني). و تنقسم هذه الخطوات إلى خطوات حساب و خطوات تواصل . و يعتبر زمن التواصل أساسي في الخوارزميات الموزعة لأنه مرتبط بخصائص شبكة الربط و عدد و حجم الرسائل المتبادلة بين المعالجات أثناء تنفيذ الخوارزمي. و يمكن حساب كلفة الخوارزمية المتوازية على أنها حاصل ضرب المعيارين السابقين[11] :

الكلفة ((c(n)= زمن التنفيذ ((t(n)) x عدد المعالجات المستخدمة ((p(n)).

و يمكن تصنيف الخوارزميات إلى أقسام أساسية من حيث التعقيد الزمني إلى الآتي:

– تعقيد ثابت و يرمز له بالدالة $O(1)$ ، – تعقيد لوغرتمي $O(\log(n))$، – تعقيد خطي $O(n)$

– تعقيد كثير الحدود $O(n^p)$ ، –تعقيد أسي $O(2^n)$ ، – تعقيد عاملي $O(n!)$

حيث: O (تقرأ بالأنجليزية big O : و هو رمز من بين زموز تعقيد الخوارزميات للباحث Landau)[12].

2. 4. 4 التسريع (speedup): تقاس نجاعة الخوارزميات التسلسلية بزمن تنفيذها بدلالة حجم مدخلاتها (n). لتكن $t_{seq}(n)$ هو زمن تنفيذ أفضل خوارزمي تسلسلي معروف لحل مسألة P بحجم مدخلات n. ليكن A الخوارزمي المتوازي لحل المسألة P و $t_p(n)$ زمن تنفيذه على حاسوب متوازي من p معالج. يمنكن تعريف التسريع S_p من خلال العبارة التالية:[13]

$$S_p(n)=t_{seq}(n) / t_p(n)$$

[11]:نغم ثروت سعيد ، " **الإخفاء باعتماد الخوارزميات المتوازية** " ، قسم علوم الحاسب / كلية التربية ، مجلة التربية والعلوم المجلد (25) ، العدد (1) ، جامعة الموصل ص145 ، 2011.

[12]: Rédha LOUCIF ، مصدر سبق ذكره ، صص42-44

[13] :Pierre Delisle, "Parallelisation d'un Algorithme d'Optimisation par colonie deFourmis", pp23-24.

يكون التسريع في أقصى قيمة له، عندما تكون تساوي عدد المعالجات p وهذا يعني أن توزيع تنفيذ الخوارزمي المتوازي A قد تم بصفة جيدة و لم ينتج عنه أي كلفة إضافية و بالتالي يمكن ملاحظة أن التسريع مرتبط كذلك بعدد المعالجات و أن قيمته محدودة.

3-4-4 **الفعالية (efficiency)**: تعتبر الفعالية E_p معيار آخر لقياس نجاعة الخوارزميات المتوازية و يمكن تعريفها من خلال العبارة التالية:[14]

$$E_p(n)= t_1(n) / pt_p(n)$$

أين:

$t_1(n)$: هو زمن تنفيذ الخوارزمي المتوازي عندما يكون عدد المعالجات p يساوي 1(ليس بالضرورة يساوي t_{seq}).

هذا القياس يعطي إشارة عن فعالية استعمال p معالج في تنفيذ الخوارزمي المتوازي. فإذا كانت قيمة $E_p(n)$ تساوي واحد لأي عدد من المعالجات p، هذا يدل بأن الخوارزمي المتوازي يكون p مرة أسرع تنفيذا باستعمال p معالج على أن ينفذ على معالج واحد. فكلما كانت قيمة الفعالية كبيرة فكلما كان الحل المتوازي أفضل. ومن جهة أخرى و لأسباب اقتصادية، فإن عدد المعالجات الذي يتطلبه تنفيذ خوارزمي متوازي له أهمية معتبرة. فإذا افترضنا على سبيل المثال خوارزميين لهما نفس زمن التنفيذ ، فإن الخوارزمي الذي يتطلب أقل عدد من المعالجات في تنفيذه يكون أقل كلفة و بالتالي هو الأفضل.[15]

4-4-4 **معيار قابلية التمديد (Iso-efficiency/scalability)** : يتمثل هذا المعيار في الخوارزمي المتوازي في قياس كمية العمل الإضافي الضروري لضمان الفعالية المتوازية عند ارتفاع عدد المعالجات. في بعض الأحيان، تكون فعالية الخوارزمية المتوازية جيدة و لكن قد تنخفض عندما يرتفع عدد المعالجات المستعملة عند تنفيذ الخوارزمي . و بالتالي فهذا المعيار يمثل العلاقة التي يجب أن تكون بين عدد المعالجات (p) و حجم المدخلات (n) قصد الحصول على فعالية معينة.[16]

4-4- 3 **مقارنة نجاعة خوارزميات الفرز**: هناك بصفة عامة طريقتين لمقارنة فعالية الخوارزميات:

[14]: Rédha LOUCIF ، مصدر سبق ذكره ، ص25

[15]: Pierre Delisle، مصدر سبق ذكره ، صص42-44

[16]: Rédha LOUCIF ، مصدر سبق ذكره ، ص42

أ- المقارنة النظرية و التي تعتمد على حساب دالة التعقيد الزمني(complexity).

ب- المقارنة التجريبية و التي تعتمد على حساب زمن التنفيذ(runtime).

ولإعطاء نظرة حول مختلف التعقيدات، أنواعها، وزمن تنفيذها لبعض خوارزميات الفرز المعروفة، نوضح ذلك من خلال الجداول التالية:

أ- المقارنة النظرية لخوارزميات الفرز التسلسلية البسيطة: نعتمد في المقارنة على التحليل النظري لدوال التعقيد الخاصة بهذه الخوارزميات

الجدول رقم (4-1): دالة التعقيد حسب عدد عمليات المقارنة

الخوارزمي	أفضل حالة(best-case)	النوع	أسوأ حالة(worst-case)	النوع
الفرز الاختياري	$O(n^2)$	تربيعي	$O(n^2)$	تربيعي
الفرز الفقاعي	$O(n)$	خطي	$O(n^2)$	تربيعي
الفرز بالإدراج	$O(n)$	خطي	$O(n^2)$	تربيعي

المصدر: R. Dumont, "Algorithme P2: La complexité ", 2009, pp. 2-25

نلاحظ من خلال الجدول (4-1) أن تعقيد جميع الخوارزميات المذكورة تربيعي في أسوأ الأحوال، و بالتالي ليس هناك أفضلية بينهم، بينما في أفضل الأحوال نجد أن خوارزمي الفرز الاختياري يكون غير محبد لأن تعقيده تربيعي و بالتالي يتطلب وقت تنفيذ أكبر من خوارزمي الفرز الفقاعي و خوارزمي الفرز بالإدراج. في الجدول الموالي نعطي ملخص لدالة التعقيد خوارزميات الفرز التسلسلية البسيطة حسب عملية التبديل.

الجدول رقم (4−2) : دالة التعقيد حسب عدد عمليات التبديل.

الخوارزمي	أفضل حالة	النوع	أسوأ حالة	النوع
الفرز الاختياري	O(n)	خطي	O(n)	خطي
الفرز الفقاعي	O(1)	ثابت	$O(n^2)$	تربيعي
الفرز بالإدراج	O(n)	خطي	$O(n^2)$	تربيعي

المصدر: Karim Baina,"Programmation avancée",ENSIAS−Rabat(Maroc), https://www.youtube.com/watch?v=X37E1wAT5Wg

في الجدول (4−2)، نلاحظ أن الفرز الفقاعي له تعقيد ثابت مهما تكن قيمة المدخل n في أفضل الأحوال و بالتالي له زمن تنفيذ أقل من الخوارزميات الأخرى، و بالتالي يكون هو المحبذ. بينما في أسوأ الأحوال فإن خوارزمي الفرز الاختياري له تعقيد خطي و بالتالي يكون هو الأفضل من ناحية زمن التنفيذ.

الجدول رقم (4−3) : دالة التعقيد حسب عمليات المقارنة.

الخوارزمي	أفضل حالة	النوع	أسوأ حالة	النوع
الفرز الاختياري)	$O(n^2)$	تربيعي	$O(n^2)$	تربيعي
الفرز الفقاعي	O(n)	خطي	$O(n^2)$	تربيعي
الفرز بالإدراج	O(n)	خطي	$O(n^2)$	تربيعي
الفرز السريع	O(n log n)	لوغريتمي	$O(n^2)$	تربيعي
الفرز بالتقسيم و الدمج	O(n log n)	لوغريتمي	O(n log n)	لوغريتمي
الفرز الكومي	O(n log n)	لوغريتمي	O(n log n)	لوغريتمي

المصدر: Eric Trichet, "**Introduction à la complexité algorithmique** ", Université Limoges,2015,pp.13−14

في الجدول (4-3) ، نلاحظ أن كل من الخوارزميات الفرز الفقاعي ، الاختياري و بالإدراج لها تعقيد تربيعي في أسوأ الأحوال. و أن خوارزمي الفرز بالإدراج هو الأسوأ بالنسبة لعدد التبديلات في أسوأ الأحوال. و أن خوارزمي الفرز الاختياري هو أسوأ خوارزميات الفرز بالنسبة لعدد عمليات المقارنة.

ب- المقارنة التجريبية:

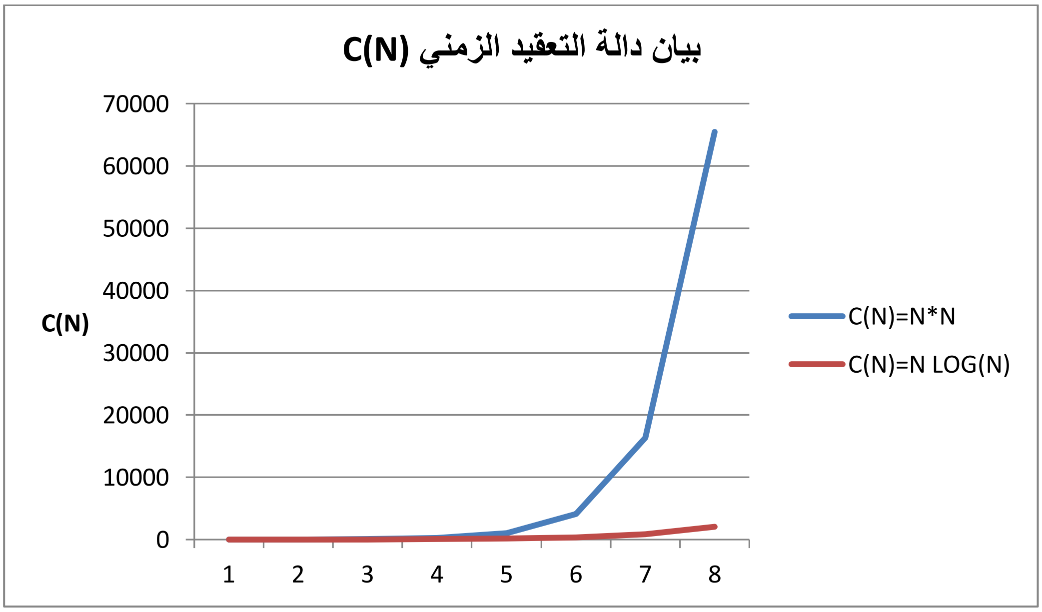
في المقارنة التجريبية ، لا نكتفي بالتحليل النظري فقط لعبارات دوال التعقيد و إنما لحسابها ثم بعد ذلك تتم المقارنة بينهم و هذا ما يوضحه الجدول(4-4).

الجدول رقم (4-4) : المقارنة التجريبية لدالة التعقيد الزمني C(N) حسب حجم الإدخال N :

N	**C(N)=N*N**	**C(N)=N LOG(N)**	**speedup(التسريع)**
2	4	2	50%
4	16	8	50%
8	64	24	63%
16	256	64	75%
32	1024	160	84%
64	4096	384	91%
128	16384	896	95%
256	65536	2048	97%

المصدر: مخرجات برنامج إكسل (Excel)

الشكل (4-6): رسم بياني لدالة التعقيد الزمني C(N)



المصدر: مخرجات برنامج إكسل (Excel)

نلاحظ في الجدول(4-4) و في الشكل (4-6) :- أن الخوارزميات ذات التعقيد C(N)= NLOG(N) لها أقل زمن تنفيذ (runtime) من الخوارزميات ذات التعقيد التربيعي C(N)=N*N.

- نلاحظ كذلك أنه كلما ارتفعت قيمة N كلما زادت نسبة التسريع(من 50% إلى 97%) .

- الخوارزميات ذات التعقيد التربيعي ليست ذات أهمية بالنسبة للقوائم من الحجم الكبير و تتطلب زمن تنفيذ كبير جدا. و يضح هذا خاصة ابتداء من النقطة (N=64) .

- الخوارزميات المتقدمة مثل(الفرز السريع ، الفرز بالدمج ، الفرز الكومي) تصبح ضرورية لمثل هذه القوائم كبيرة الحجم.

ج - المقارنة النظرية لخوارزميات الفرز المتوازية :

نقوم في ما يلي بعرض دالة التعقيد الزمني Tp(n, p)، التسريع Sp(n, p) و الفعالية Ep(n, p) لبعض خوارزميات الفرز المتوازية (n هو حجم المعطيات و p هو عدد المعالجات).

الجدول (4-5) : مقارنة جودة خوارزميات الفرز المتوازية.

الخوارزمي	التعقيد الزمني TS	التعقيد الزمني Tp	التسريع Sp	الفعالية Ep
الفرز الفردي-الزوجي (odd even sort)	$n\log_2 n$	$(n/p)\log_2(n/p) + 2n$	Ts/Tp	Sp/p
الفرز (shell sort)	$n\log_2 n$	$(n/p)\log_2(n/p) + 2n$	Ts/Tp	Sp/p
الفرز السريع (quick sort)	$n\log_2 n$	$(n/p)\log_2(n/p) + \log_2 p *(2n/p)$	Ts/Tp	Sp/p
الفرز بالدمج (merge sort)	$n\log_2 n$	$(n/p)\log_2(n/p) + \log_2 p *(2n/p)$	Ts/Tp	Sp/p

المصدر: Gergel V. P., "Introduction to parallel programming: Parallel Methods for sorting", pp.10-30

نلاحظ من خلال الجدول رقم(4-5) أن جميع خوارزميات الفرز المذكورة لها نفس التعقيد في صيغتها المتسلسلة Ts=nlogn و هو لوغريتمي ولكن الاختلاف موجود في التعقيد الزمني المتوازي Tp. و يتجلى هذا بين خوارزمي الفرز الفردي-الزوجي المتوازي من جهة وخوارزمي الفرز السريع و خوارزمي الفرز بالدمجع المتوازيان من جهة أخرى.

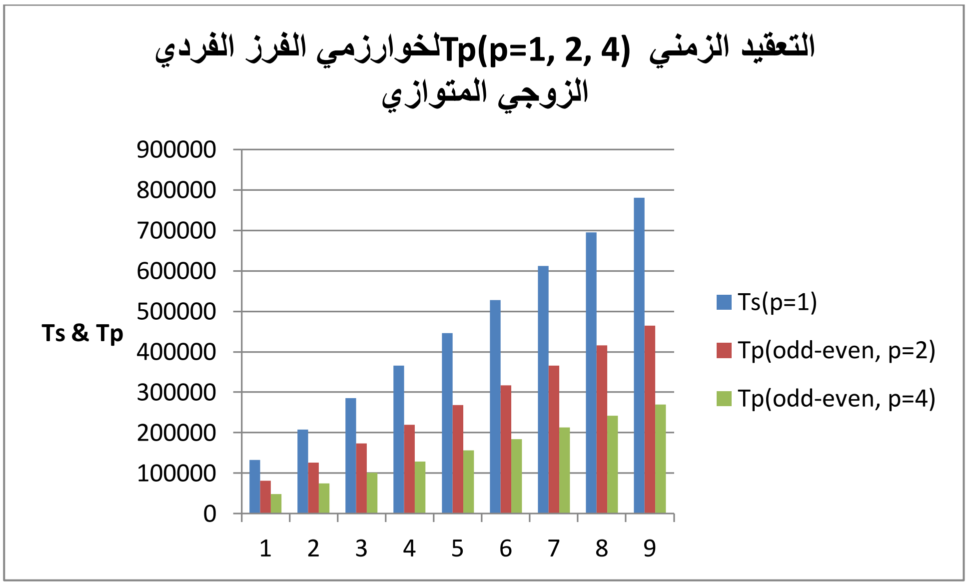
و قصد إجراء مقارنة تجريبية لنجاعة خوارزميات الفرز المذكورة في الجدول(4-5) سواء كان هذا بين صيغها التسلسلية و المتوازية من جهة أو فيما بينها ، سنقوم بحساب معايير النجاعة المتمثلة في التعقيد الزمني ، التسريع و الفعالية لكل خوارزمي و بالاستعانة ببرنامج EXCEL كما هو مبين في الجدول(4-6) بالنسبة لخوارزمي الفرز المتوازي الفردي-الزوجي و بنفس طريقة الحساب مع خوارزمي الفرز السريع المتوازي في الجدول(4-7) .

الجدول (4-6) : حساب التعقيد الزمني ((Tp(p=2; p=4 لخوارزمي الفرز المتوازي الفردي-الزوجي

حساب التعقيد الزمني لخوارزمي الفرز المتوازي الفردي الزوجي							
size n	Ts	Tp(p=2)	Tp(p=4)	Sp(p=2)	Sp(p=4)	Ep(p=2)	Ep(p=4)
10000	132877,1238	81438,5619	48219,28095	1,631624143	2,755684473	0,815812072	0,688921118
15000	208090,1232	126545,0616	74522,5308	1,644395448	2,792311546	0,822197724	0,698077887
20000	285754,2476	172877,1238	101438,5619	1,65293268	2,817017929	0,82646634	0,704254482
25000	365241,0119	220120,5059	128810,253	1,659277541	2,835496426	0,829638771	0,708874106
30000	446180,2464	268090,1232	156545,0616	1,664291997	2,85017133	0,832145998	0,712542832
35000	528327,3556	316663,6778	184581,8389	1,668417923	2,862293272	0,834208961	0,715573318
40000	611508,4952	365754,2476	212877,1238	1,671910851	2,872589052	0,835955425	0,718147263
45000	695593,6821	415296,8411	241398,4205	1,67493131	2,881517123	0,837465655	0,720379281
50000	780482,0237	465241,0119	270120,5059	1,677586463	2,889384577	0,838793232	0,722346144

المصدر : مخرجات برنامج إكسل (Excel)

الشكل (4-7): التعقيد الزمني (Tp(p=1, 2, 4 لخوارزمي الفرز الفردي-الزوجي المتوازي.

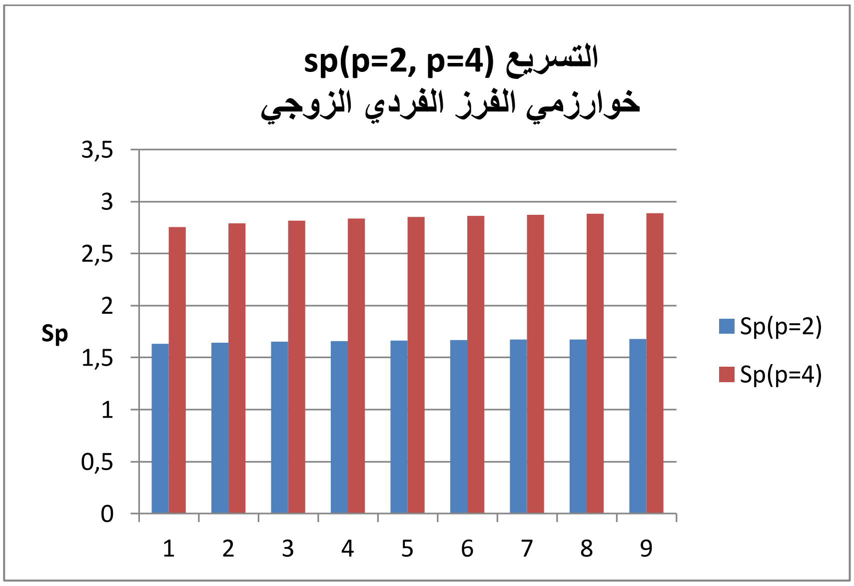


المصدر: مخرجات برنامج إكسل (Excel)

نلاحظ في الشكل (4-7) أن التعقيد الزمني لخوارزمي الفرز الفردي-الزوجي في صيغته التسلسلية يقارب تنفيذ 800000 عملية و ينخفظ هذا العدد من العمليات في صيغة الخوارزمي المتوازية إلى حوالي

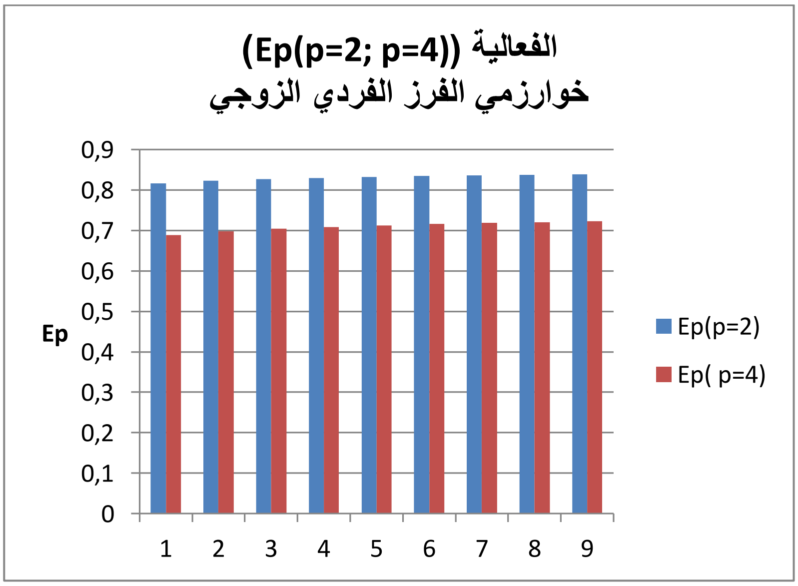
500000 عملية عندما يكون عدد المعالجات إثنان (p=2) و إلى حوالي 300000 عملية عندما يكون عدد المعالجات أربعة (p=4) و بالتالي انخفاظ زمن التنفيذ.

شكل (4-8): التسريع Sp(p=2, p=4) لخوارزمي الفرز المتوازي الفردي-الزوجي



نلاحظ في الشكل رقم(4-8) أن التسريع Sp يتضاعف مع عدد المعالجات p و هذا لكل أحجام المعطيات N.

شكل(4-9): الفعالية Ep(p=2, p=4) لخوارزمي الفرز المتوازي الفردي-الزوجي



المصدر: مخرجات برنامج إكسل (Excel)

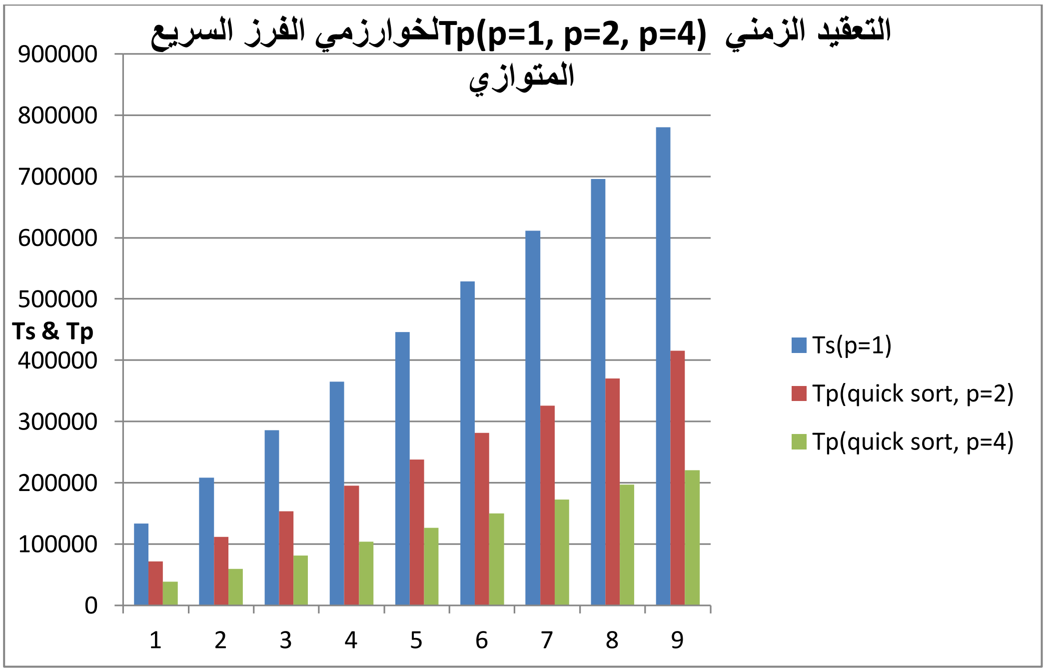
في الشكل رقم (4-9) الفعالية Ep تقارب 0.9 عندما يكون عدد المعالجات p=2 وتقارب 0.7 عندما يكون عدد المعالجات p=4 و بالتالي فالزيادة في عدد المعالجات في تنفيذ الخوارزمي غير مجدية.

الجدول رقم(4-7) : حساب التعقيد الزمني ((Tp(p=2; p=4) لخوارزمي الفرز السريع المتوازي

حساب التعقيد الزمني ، التسريع و الفعالية لخوارزمي الفرز السريع المتوازي (parallel quick sort)							
size n	**Ts(p=1)**	**Tp(p=2)**	**SP(p=2)**	**Ep(p=2)**	**Tp(p=4)**	**Sp(p=4)**	**Ep(p=4)**
10000	132877,124	71438,5619	1,86001958	0,9300098	38219,2809	3,476703917	0,86917598
15000	208090,123	111545,0616	1,8655252	0,9327626	59522,5308	3,495989173	0,87399729
20000	285754,248	152877,1238	1,86917598	0,934588	81438,5619	3,508832191	0,87720805
25000	365241,012	195120,5059	1,87187405	0,935937	103810,253	3,518352007	0,879588
30000	446180,246	238090,1232	1,87399729	0,9369986	126545,062	3,525860597	0,88146515
35000	528327,356	281663,6778	1,87573833	0,9378692	149581,839	3,532028751	0,88300719
40000	611508,495	325754,2476	1,87720805	0,938604	172877,124	3,537243574	0,88431089
45000	695593,682	370296,8411	1,87847587	0,9392379	196398,421	3,541747842	0,88543696
50000	780482,024	415241,0119	1,879588	0,939794	220120,506	3,545703388	0,88642585

المصدر: مخرجات برنامج إكسل (Excel)

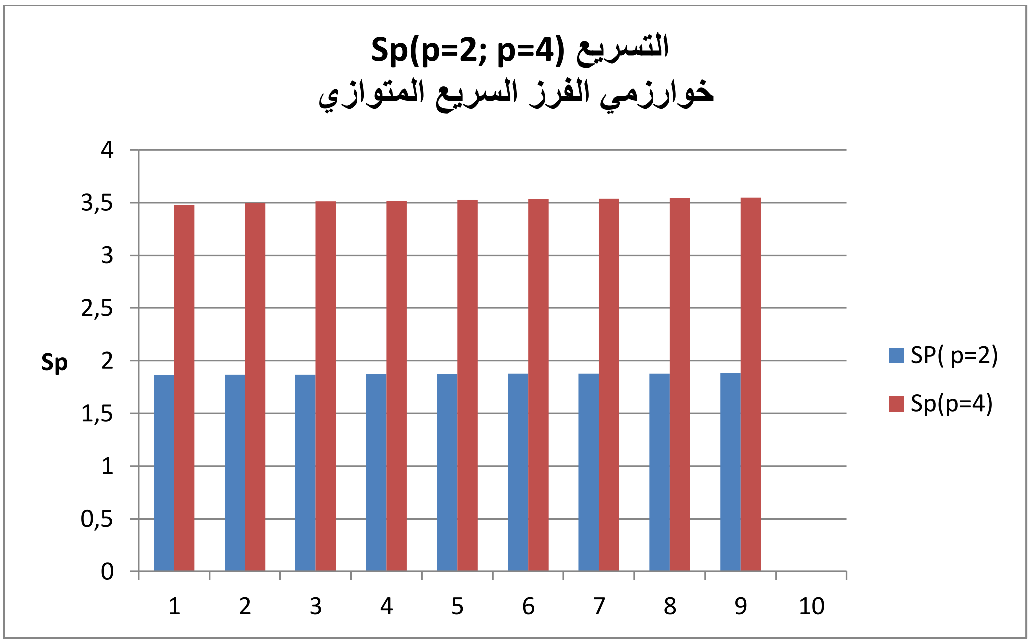
الشكل (4-10): التعقيد الزمني (Tp(p=1, 2, 4 لخوارزمي الفرز السريع المتوازي



المصدر: مخرجات برنامج إكسل (Excel)

نلاحظ كذلك في الشكل(4−10) أن زمن تنفيذ خوارزمي الفرز السريع المتسلسل يقارب تنفيذ 800000 عملية و ينخفض إلى حوالي 50% في صيغة الخوارزمي المتوازية عندما يكون عدد المعالجات إثنان(p=2) و إلى 25% عندما يكون عدد المعالجات أربعة (p=4).

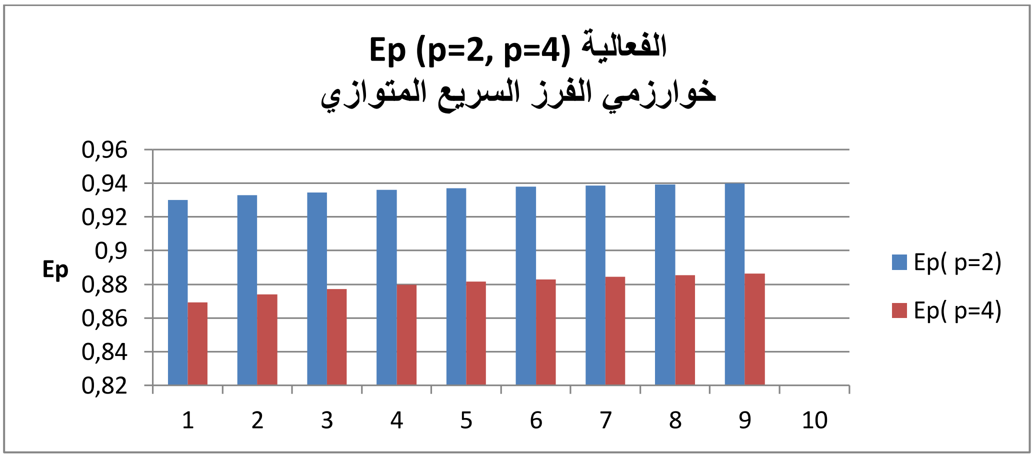
شكل (4−11): التسريع Sp(p=2, p=4) لخوارزمي الفرز السريع المتوازي



<u>المصدر</u>: مخرجات برنامج إكسل (Excel)

في الشكل (4−11) نلاحظ أن التسريع Sp(p=4) يساوي حوالي 3.5 عندما يكون عدد المعالجات p=4 و يقارب 2 عندما يكون عدد المعالجات (p=2) و بالتالي فالتسريع يتضاعف مع تضاعف عدد المعالجات التي يتم تنفيذ الخوارزمي عليها.

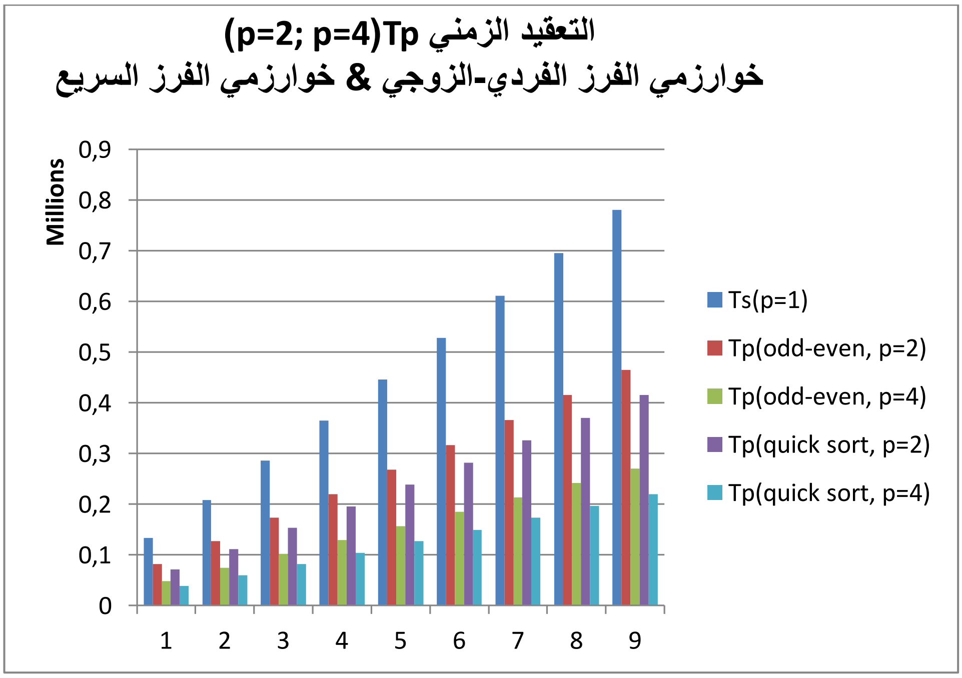
الشكل (4-12): الفعالية Ep(p=2, p=4) لخوارزمي الفرز السريع المتوازي



المصدر: مخرجات برنامج إكسل (Excel)

في الشكل (4-12) نلاحظ أن فعالية الخوارزمي Ep(p=4) تساوي حوالي0.88 و تقارب 0.94 عندما يكون عدد المعالجات (p=2) و بالتالي فزيادة عدد المعالجات في تنفيذ خوارزمي الفرز السريع غير مجد.

الشكل (4-13): التعقيد الزمني Tp(p=1, 2, 4) لخوارزمي الفرز الفردي-الزوجي المتوازي خوارزمي الفرز السريع المتوازي.



المصدر: مخرجات برنامج إكسل (Excel)

نلاحظ في البيان (4-13) أن التعقيد الزمني Tp في خوارزمي الفرز السريع أقل منه في خوارزمي الفردي- الزوجي و هذا سواء تم تنفيذ الخوارزمي على حاسوب من معالجين (p=2) أو أربع معالجات (p=4) وبالتالي خوارزمي الفرز السريع هو الأفضل من ناحية زمن التنفيذ. نلاحظ كطلك أنه كلما أرتفع عدد المعالجات من 2 إلى 4 نقص زمن التنفيذ Tp

4-5 مقارنة الخوارزميات في بحوث العمليات(نظرية البيان)

1: الخوارزميات المتسلسلة و المتوازية لإيجاد الشجرة بأقل تغطية في بيان G(V,E)

أين : V : مجموعة القمم في البيان (|V|:عدد القمم) E : مجموعة الأضلاع في البيان (|E| :عدد الأضلاع)

نقوم في ما يلي بمقارنة فعالية كل من خوارزمي Prim و خوارزمي Kruskal و هذا من خلال معيار التعقيد الزمني التسلسلي Ts و التعقيد الزمني المتوازي Tp(p: يمثل عدد المعالجات أو النواة في الحاسوب).

الجدول (4-8): التعقيد الزمني لخوارزمي Prim و خوارزمي Kruskal.

الخوارزمي	التعقيد الزمني	
	التعقيد الزمني المتسلسل(Ts)	التعقيد الزمني المتوازي(Tp)
خوارزمي (Prim)	$O(\|V\|x\|E\|)=O(n^2)$	$O(n^2/p)+O(n \log p)$
خوارزمي (Kruskal)	$O(\|V\|\log(\|V\|)=O(n \log n)$	$O(n^2/p)+O(n \log p)$

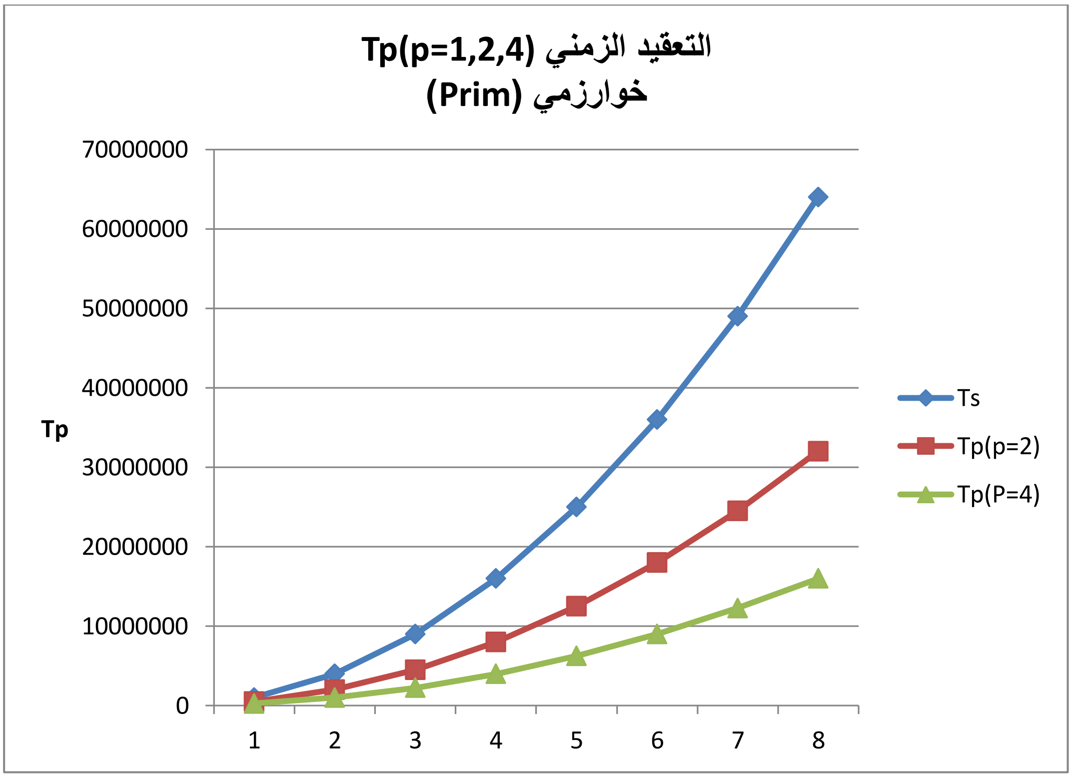
المصدر Gergel V. P., "Parallel Methods For Graph Calculations: "parallel Graph Algorithms",pp.2-11

نقوم في الجدول(4-9) و الجدول (4-11)بحساب التعقيد الزمني لكل من خوارزمي Prim و Kruskal على التوالي اعتمادا على دوال التعقيد المدونة في الجدول(4-8) وباستعمال برنامج EXCEL.

الجدول (4-9): حساب التعقيد الزمنب لخوارزمي بريم حساب التعقيد الزمني لخوارزمي Prim المتسلسل Ts و المتوازي Tp(p=2, p=4)							
size n	Ts	T(calc)(p=2)	T(com)(p=2)	Tp(p=2)	T(calc)(p=4)	T(com)(p=4)	Tp(P=4)
1000	1000000	500000	1000	501000	250000	2000	252000
2000	4000000	2000000	2000	2002000	1000000	4000	1004000
3000	9000000	4500000	3000	4503000	2250000	6000	2256000
4000	16000000	8000000	4000	8004000	4000000	8000	4008000
5000	25000000	12500000	5000	12505000	6250000	10000	6260000
6000	36000000	18000000	6000	18006000	9000000	12000	9012000
7000	49000000	24500000	7000	24507000	12250000	14000	12264000
8000	64000000	32000000	8000	32008000	16000000	16000	16016000

المصدر: مخرجات برنامج إكسل (Excel)

الشكل (4-14): حساب التعقيد الزمني ((Tp(p=2; p=4) لخوارزمي (Prim)



المصدر: مخرجات برنامج إكسل (Excel)

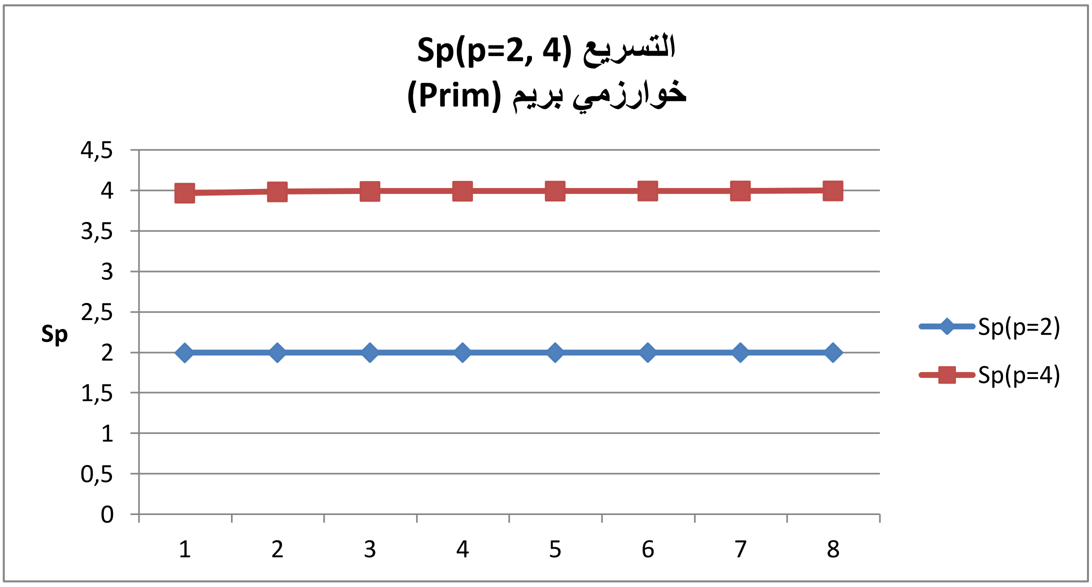
نلاحظ في الشكل (4-14) أن أدنى تعقيد الزمني متوازي Tp لخوارزمي بريم يتحقق عندما يكون عدد المعالجات أربعة (p=4) و يليه التعقيد الزمني المتوازي Tp(p=2) في حين يسجل التعقيد المتسلسل أعلى قيمة و هذا كلما ازداد حجم المدخلات أو المعطيات N.

الجدول (4-10) : حساب التسريع Sp(p=2,p= 4) و الفعالية Ep(p=2,p= 4) لخوارزمي بريم

حساب التسريع Sp و الفعالية Ep لخوارزمي بريم المتوازي				
Size n	**Sp(p=2)**	**Sp(p=4)**	**Ep(p=2)**	**Ep(p=4)**
1000	1,99600798	3,96825397	0,99800399	0,99206349
2000	1,998002	3,98406375	0,999001	0,99601594
3000	1,99866755	3,9893617	0,99933378	0,99734043
4000	1,9990005	3,99201597	0,99950025	0,99800399
5000	1,99920032	3,99361022	0,99960016	0,99840256
6000	1,99933356	3,99467377	0,99966678	0,99866844
7000	1,99942873	3,99543379	0,99971437	0,99885845
8000	1,99950012	3,996004	0,99975006	0,999001

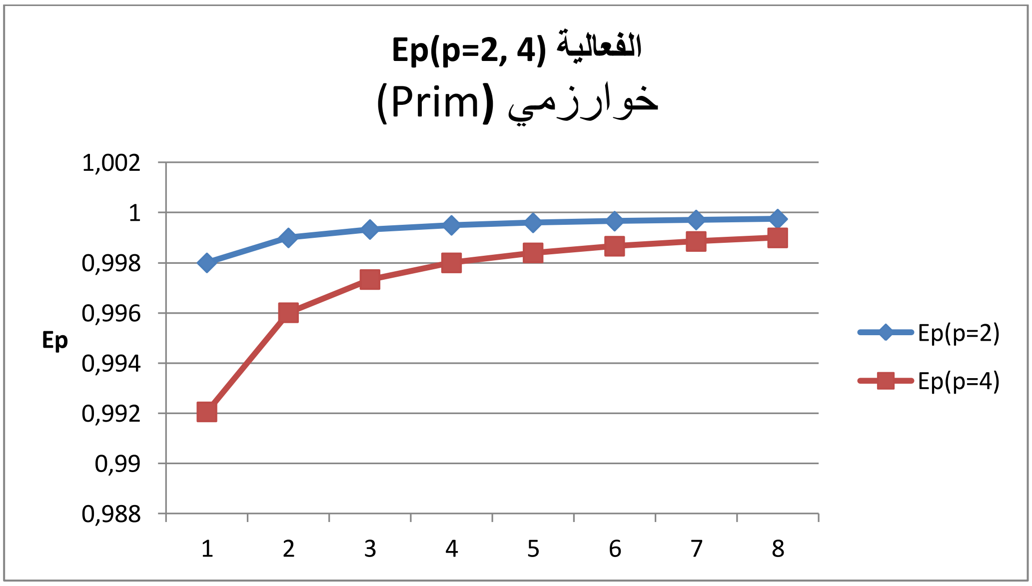
المصدر: مخرجات برنامج إكسل (Excel)

شكل (4-15): التسريع Sp(p=2, p=4) لخوارزمي بريم (Prim) المتوازي



المصدر: مخرجات برنامج إكسل (Excel)

الشكل (4-16): الفعالية Ep(p=2, p=4) لخوارزمي (Prim) المتوازي



المصدر: مخرجات برنامج إكسل (Excel)

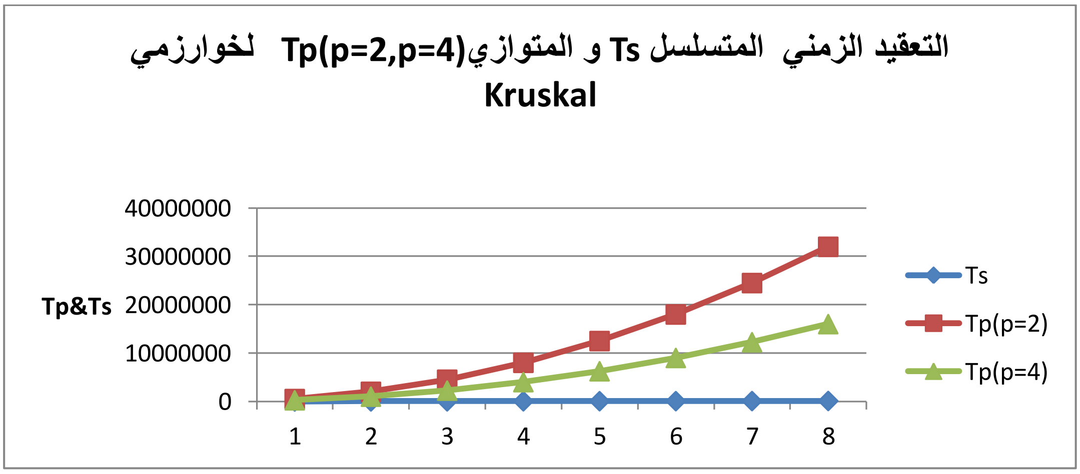
نلاحظ في الشكل(4−16) أن منحنى الفعالية لخوارزمي **Prim** المتوازي يؤول إلى واحد و هذا كلما أرتفع حجم معطيات الإدخال N و هذا سواء نفذ الخوارزمي على حاسوب من معالجين أو أربع معالجات. و بالتالي فزيادة عدد المعالجات في هذه الحالة غير مجدية .

الجدول (4−11): حساب التعقيد الزمني لخوارزمي (Kruskal) المتسلسل Ts و المتوازي Tp(p=2,p=4)

حساب التعقيد الزمني المتسلسل و المتوازي Tp(p=2, p=4) لخوارزمي كريسكال			
size n	**Ts**	**Tp(p=2)**	**Tp(p=4)**
1000	9965,78428	501000	252000
2000	21931,5686	2002000	1004000
3000	34652,2404	4503000	2256000
4000	47863,1371	8004000	4008000
5000	61438,5619	12505000	6260000
6000	75304,4807	18006000	9012000
7000	89411,9744	24507000	12264000
8000	103726,274	32008000	16016000

المصدر: مخرجات برنامج إكسل (Excel)

الشكل (4−17): حساب التعقيد الزمني المتسلسل Ts و المتوازي ((Tp(p=2; p=4) لخوارزمي Kruskal



المصدر: مخرجات برنامج إكسل (Excel)

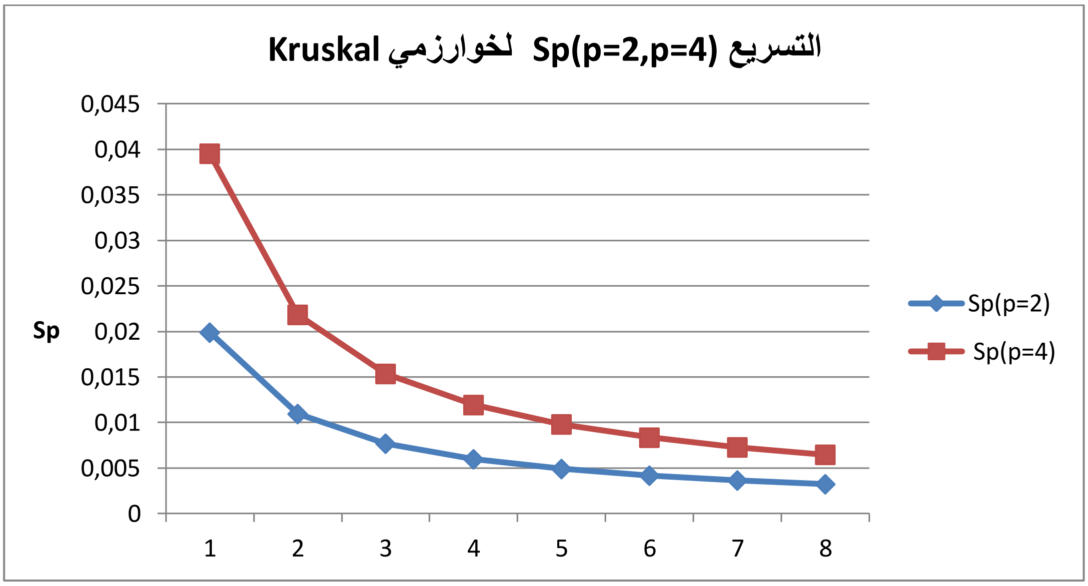
يظهر الشكل(4−17) أن التعقيد الزمني التسلسلي Ts أقل بكثير من التعقيد الزمني المتوازي Tp و يزداد هذا الفارق كلما ازداد حجم المعطيات المدخلة وهذا ما يفسر أنه ليس دائما يكون خوارزمي **Kruskal** المتوازي له أقل زمن تنفيذ.

الجدول (4−12) : حساب التسريع Sp(p=2,p= 4) و الفعالية Ep(p=2,p= 4) لخوارزمي Kruskal

حساب التسريع Sp(p=2, p=4) و الفعالية Ep (p=2,p=4) لخوارزمي كريسكال				
Size n	**Sp(p=2)**	**Ep(p=2)**	**Sp(p=4)**	**Ep(p=4)**
1000	0,01989178	0,00994589	0,03954676	0,00988669
2000	0,01095483	0,00547741	0,02184419	0,00546105
3000	0,00769537	0,00384768	0,01536004	0,00384001
4000	0,0059799	0,00298995	0,0119419	0,00298548
5000	0,00491312	0,00245656	0,00981447	0,00245362
6000	0,00418219	0,00209109	0,00835602	0,00208901
7000	0,00364843	0,00182421	0,0072906	0,00182265
8000	0,00324064	0,00162032	0,00647642	0,0016191

المصدر: مخرجات برنامج إكسل (Excel)

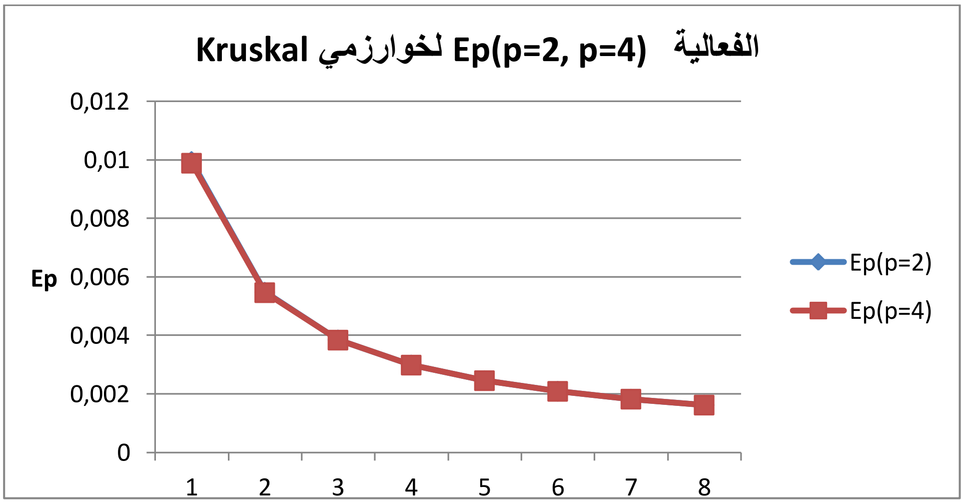
الشكل (4-18): التسريع Sp(p=2, p=4) لخوارزمي (Kruskal) المتوازي



المصدر: مخرجات برنامج إكسل (Excel)

الشكل(4-18) يظهر أن تسريع خوارزمي **Kruskal** المتوازي في تناقص و هذا كلما أرتفع حجم المعطيات سواء نفذ الخوارزمي على حاسوب من معالجين أو من أربع معالجات و بالتالي فالصيغة المتوازية للخوارزمي غير مجدي.

الشكل (4-19): افعالية Ep(p=2, p=4) لخوارزمي (Kruskal) المتوازي



المصدر: مخرجات برنامج إكسل (Excel)

2- **2: الخوارزميات المتسلسلة و المتاوزية لإيجاد أقصر مسار في البيان G(V, E)**.نلخص في الجدول الموالي دوال التعقيد الزمني لكل من خوارزمي **Dijkstra** و **Floyd** في صيغها التسلسلية و المتوازية

الجدول (4-13) : التعقيد الزمني لخوارزمي (Dijkstra) و خوارزمي (Floyd-Warshal)

الخوارزمي	**التعقيد الزمني**	
	الصيغة المتسلسلة (Ts)	**الصيغة المتوازية (Tp)**
خوارزمي Dijkstra	**$Ts=O(\lvert V\rvert^2)=O(n^2)$**	**$Tp=O(n^3/p)+O(n\log(p))$** **(computation)+(communication)**
خوارزمي Floyd-Warshal	**$Ts=O(\lvert V\rvert^3)=O(n^3)$**	**$Tp=O(n^3/p)+O(n^2/\sqrt{p}\log(p))$** **(computation)+(communication)**

المصدر 1: Gergel V. P., "Parallel Methods For Graph Calculations: "parallel Graph Algorithms", pp.2-11

المصدر 2: Vivek Sarkar, "**Parallel Graph Algorithms**", Rice University, 2008, pp.2-34

يظهر في الجدول (4-13) أن خوارزمي (Dijkstra) المتسلسل له تعقيد تربيعي بينما في صيغته المتوازية تكعيبي و بالتالي Ts أقل بكثير من Tp (Ts<<Tp) . أما خوارزمي (Floyd) فتعقيده الزمني المتسلسل Ts و المتوازي Tp متقاربان نوعا ما (Ts ≈Tp) و هو ما سنوضحه في البيانات الموالية في الجدول(4-14) بالنسبة لخوارزمي Dijkstra و الجدول(4-15) والجدول(4-16) لخوارزمي Floyd.

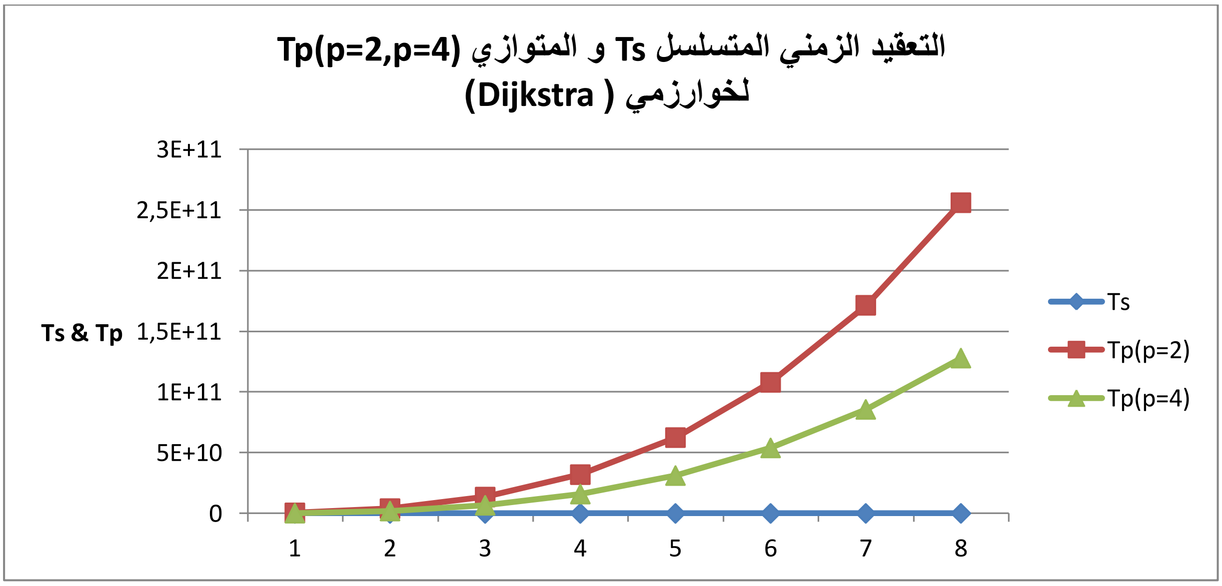
الجدول (4-14) : حساب التعقيد المتسلسل Ts والمتوازي Tp(p=2,p= 4) لخوارزمي Dijkstra

حساب التعقيد الزمني المتسلسل Ts و المتوازي Tp لخوارزمي Dijkstra							
size n	**Ts**	**Tcalc(p=2)**	**Tcom(p=2)**	**Tp(p=2)**	**Tcalc(p=4)**	**Tcom(p=4)**	**Tp(p=4)**
1000	1000000	500000000	1000	500001000	250000000	2000	250002000
2000	4000000	4000000000	2000	4000002000	2000000000	4000	2000004000
3000	9000000	13500000000	3000	1,35E+10	6750000000	6000	6750006000
4000	1,6E+07	32000000000	4000	3,2E+10	16000000000	8000	1,6E+10
5000	2,5E+07	62500000000	5000	6,25E+10	31250000000	10000	3,125E+10
6000	3,6E+07	1,08E+11	6000	1,08E+11	54000000000	12000	5,4E+10
7000	4,9E+07	1,715E+11	7000	1,715E+11	85750000000	14000	8,575E+10
8000	6,4E+07	2,56E+11	8000	2,56E+11	1,28E+11	16000	1,28E+11

المصدر: مخرجات برنامج إكسل (Excel)

الشكل (4-20) : حساب التعقيد الزمني المتسلسل Ts والمتوازي ((Tp(p=2; p=4) لخوارزمي

Dijkstra



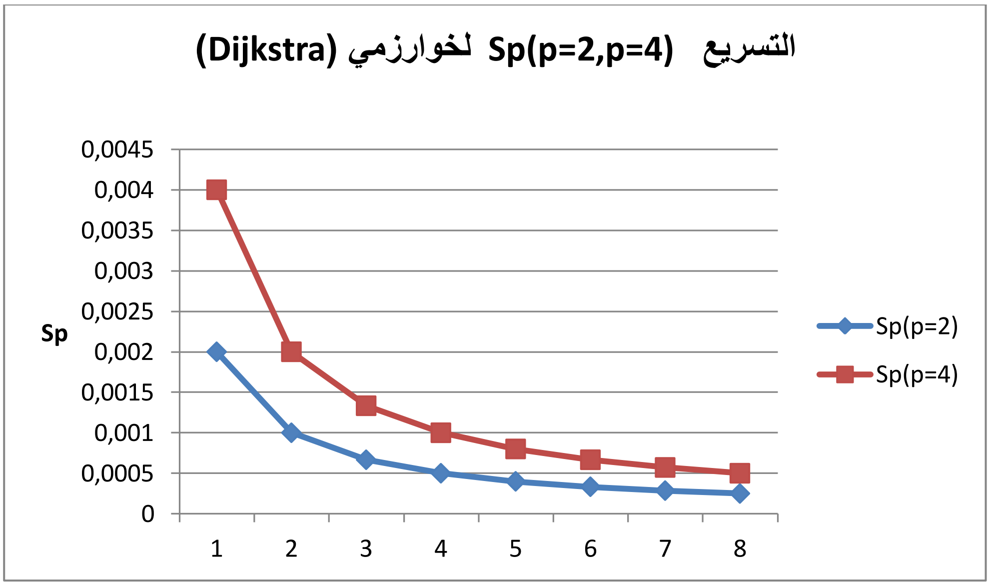
المصدر: مخرجات برنامج إكسل (Excel)

يظهر الشكل(4−20) أن التعقيد الزمني التسلسلي Ts أقل بكثير من التعقيد الزمني المتوازي Tp و يزداد هذا الفارق كلما ازداد حجم المعطيات المدخلة وهذا ما يفسر أنه ليس دائما يكون خوارزمي المتوازي له أقل زمن تنفيذ.

الجدول (4−15) : حساب التسريع Sp(p=2,p= 4) و الفعالية Ep(p=2,p= 4) لخوارزمي Dijkstra

حساب التسريع Sp(p =2, p=4) و الفعالية Ep(p=2, p=4) لخوارزمي Dijkstra				
Size n	**Sp(p=2)**	**Sp(p=4)**	**Ep(p=2)**	**Ep(p=4)**
1000	0,002	0,00399997	0,001	0,00099999
2000	0,001	0,002	0,0005	0,0005
3000	0,00066667	0,00133333	0,00033333	0,00033333
4000	0,0005	0,001	0,00025	0,00025
5000	0,0004	0,0008	0,0002	0,0002
6000	0,00033333	0,00066667	0,00016667	0,00016667
7000	0,00028571	0,00057143	0,00014286	0,00014286
8000	0,00025	0,0005	0,000125	0,000125

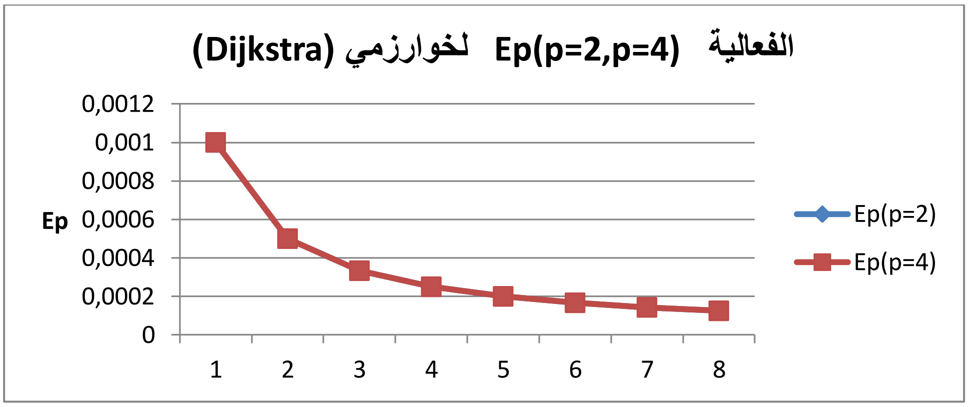
الشكل (4−21): التسريع Sp(p=2, p=4) لخوارزمي (Dijkstra)



المصدر: مخرجات برنامج إكسل (Excel)

نلاحظ في الشكل(4-21) أن التسريع Sp في تناقص سواء كان عدد المعالجات p=2 أو p=4 و هذا كلما أرتفع حجم المعطيات N و بالتالي فزيادة عدد المعالجات لتنفيذ الخوارزمي Dijkstra ليس له تأثير على التسريع و خاصة عندما يكون حجم المعطيات كبير جدا.

الشكل (4-22): الفعالية Ep(p=2, p=4) لخوارزمي (Dijkstra)



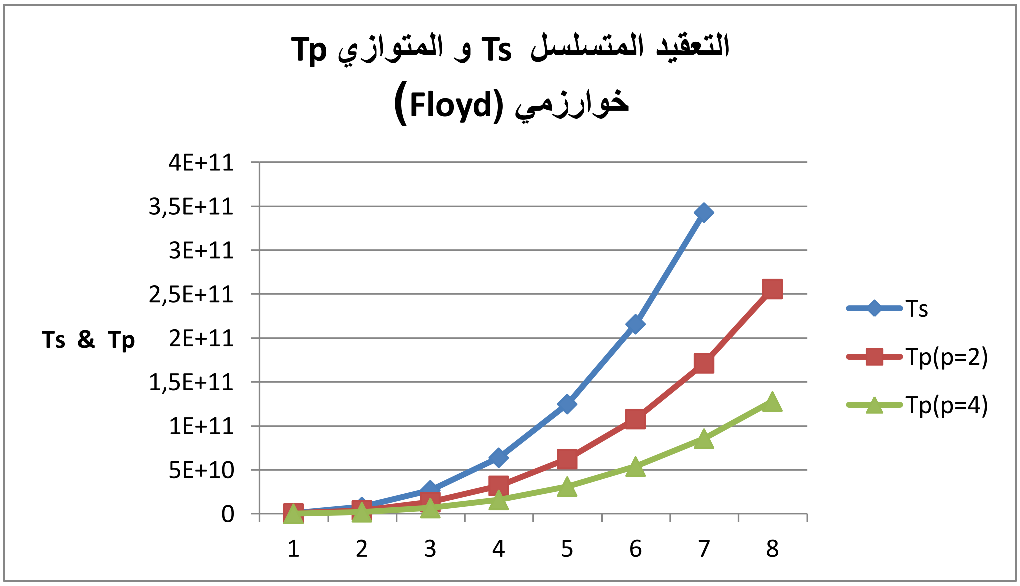
المصدر : مخرجات برنامج إكسل (Excel)

الشكل (4-22) يظهر أن فعالية خوارزمي Dijkstra في تناقص و هذا كلما ارتفع حجم المعطيات سواء تم تنفيذ الخوارزمي على معالجين p=2 أو أربع معالجات p=4 و بالتالي فالخوارزمي المتوازي ل Dijkstra غير فعال.

الجدول (4-16): حساب التعقيد المتسلسل Ts والمتوازي Tp(p=2,p= 4) لخوارزمي فلايد

حساب التعقيد المتسلسل Ts و المتوازي لخوارزمي Floyd : مخرجات برنامج إكسل (Excel)			
size n	**Ts**	**Tp(p=2)**	**Tp(p=4)**
1000	1000000000	500707107	251000000
2000	8000000000	4002828427	2004000000
3000	2,7E+10	1,3506E+10	6759000000
4000	6,4E+10	3,2011E+10	1,6016E+10
5000	1,25E+11	6,2518E+10	3,1275E+10
6000	2,16E+11	1,0803E+11	5,4036E+10
7000	3,43E+11	1,7153E+11	8,5799E+10
8000	5,12E+11	2,5605E+11	1,2806E+11
7000	3,43E+11	1,7153E+11	8,5799E+10
8000	5,12E+11	2,5605E+11	1,2806E+11

الشكل (4-23): التعقيد المتسلسل Ts و المتوازي Tp(p=2, p=4) لخوارزمي Floyd



المصدر: مخرجات برنامج إكسل (Excel)

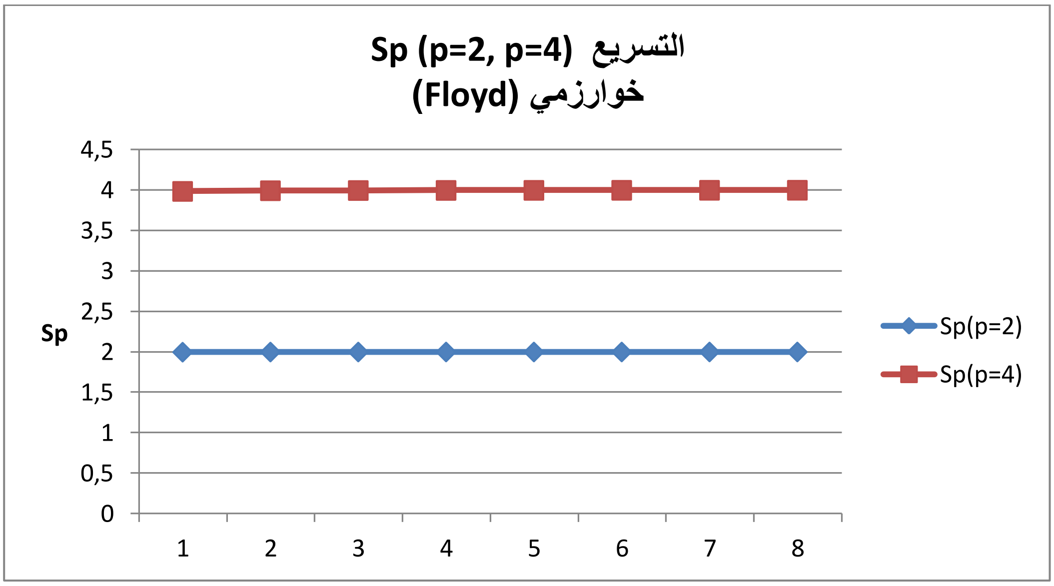
نلاحظ في الشكل (4-23) أن التعقيد الزمني المتسلسل Ts يفوق التعقيد الزمني المتوازي Tp و هذا خاصة كلما ارتفع حجم المعطيات n.

الجدول (4-17): حساب التسريع Sp(p=2,p= 4) و الفعالية Ep(p=2,p= 4) لخوارزمي Floyd

حساب التسريع Sp(p=2, p=4) و الفعالية Ep(p=2, p=4) لخوارزمي فلايد Floyd				
Size n	**Sp(p=2)**	**Sp(p=4)**	**Ep(p=2)**	**Ep(p=4)**
1000	1,99717557	3,98406375	0,99858778	0,99601594
2000	1,99858679	3,99201597	0,99929339	0,99800399
3000	1,99905764	3,99467377	0,99952882	0,99866844
4000	1,99929314	3,996004	0,99964657	0,999001
5000	1,99943447	3,99680256	0,99971724	0,99920064
6000	1,99952871	3,99733511	0,99976435	0,99933378
7000	1,99959602	3,99771559	0,99979801	0,9994289
8000	1,99964651	3,998001	0,99982325	0,99950025

المصدر: مخرجات برنامج إكسل (Excel)

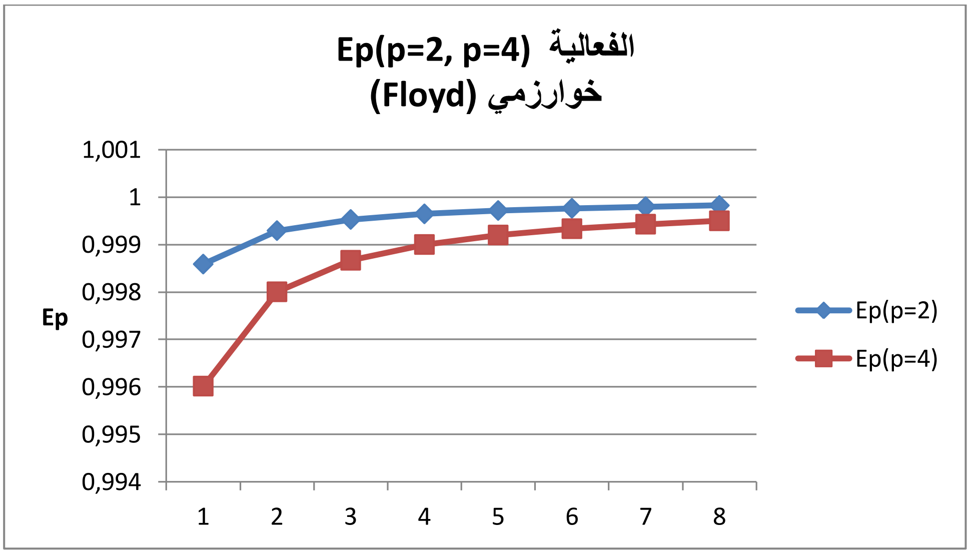
الشكل (4-24): التسريع Sp(p=2, p=4) لخوارزمي (Floyd)



المصدر: مخرجات برنامج إكسل (Excel)

نلاحظ في الشكل(4-24) أن نسبة التسريع عندما يكون عدد المعالجات p=2 يساوي 2 و يساوي 4 عندما يكون عدد المعالجات p=4 و بالتالي نقول أن التسريع يتضاعف بتضاعف المعالجات.

الشكل (4-25): الفعالية Ep(p=2, p=4) لخوارزمي (Floyd)



المصدر: مخرجات برنامج إكسل (Excel)

يظهر الشكل(4−25) أن فعالية الخوارزمي (Ep(p=4 أقل منها عندما يكون عدد المعالجات (p=2) و هذا خاصة عندما يكون حجم المعطيات (المدخلات) صغيرا نسبيا و تبدأ في التقارب كلما ارتفع حجم المعطيات حتى تؤول إلى القيمة واحد. بمعنى تكون لخوارزمي **Floyd** نفس الفعالية سواء تم تنفيذ الخوارزمي على معالجين أو أربع معالجات.

6.4 برمجة الخوارزميات المتوازية:

كما رأينا في فصل سابق أن هناك تقنيات يمكن إتباعها لجعل الخوارزمية أو مسألة معينة متوازية، لكن يبقى لدينا السؤال التالي: "كيف نقوم ببرمجة هده الخوارزمية على الحاسوب المتوازي؟"، و الإجابة: " يمكننا ذلك باستخدام نوع من البرمجة تسمى بالبرمجة المتوازية[17].

1.6.4 البرمجة المتوازية: هي البرمجة بلغة تتضمن البنى أو الميزات المتوازية.

يمكن أن يتم بناء المميزات المتوازية من خلال بعض اللغات البرمجية التي تعتمد مبدأ التوازي في تصميمها مثل.

CSP(Communicating Sequentiel Process), OCCAM

أو يمكن توسيع لغات البرمجة التسلسلية من أجل إحتواء تعليمات التوازي مثل:

Parallel Fortran, Parallel Pascal, Parallel C

و يمكن كذلك إلحاق المزايا المتوازية إلى لغة تسلسلية تقليدية مثل: Fortran أو ++C/C و ذلك باستخدام روتينات المكتبات مثل مكتبة MPI أو مكتبة OpenMP أو مكتبة PVM أو Pthread(Posix (Threads وغيرها.

OpenMP و Pthreads هي مكتبات للبرمجة المتوازية خاصة بالحاسبات ذات الذاكرة المشتركة.

المكتبة (OpenMP : Open MultiProcessing) هي عبارة عن مجموعة من الأوامر (تسمح بتعريف المقاطع المتوازية، المتسلسلة أو الحرجة)، الدوال (مثلا معرفة عدد الإجرائيات المتاحة عند التنفيذ)، والمتغيرات البيئية(تكون قيمها مثبة وتأخذ بعين الاعتبار). لهذه المكتبة واجهة برمجة للغات البرمجة C/C++, Fortran ويمكن استعمالها مع كل من نظام التشغيل .(Linux, Windows)

[17] :Fayez Gebalé ; P106

Pthreads(POSIX Threads) : هي كذلك مكتبة تسمح بإنشاء و استعمال ما يسمى بالإجرائيات الخفيفة (thread) باللغة الأنجليزية.

MPI(Message Passage Processing) : هي مكتبة دوال للبرمجة المتوازية على الأجهزة ذات الذاكرة الموزعة. يعتمد برنامج MPI على النموذج SPMD أي نسخة واحدة من برنامج MPI تنفذ على كل معالج، وتستعمل دالة MPI لإرجاع رقم المعالج. المعالجات تتواصل فيما بينها بتمرير الرسائل. يكون هذا التواصل ثنائي (مرسل و مستقبل) أو يكون جماعي (مرسل و مجموعة مستقبلين). في برنامج MPI هناك كذلك موزع مسير (commutateur) يسمح بمعرفة الإجرائيات(المعالجات) النشطة أو العاملة في كل لحظة. تضع المكتبة عدة دوال في متناول المبرمج بالقيام بمختلف أنواع الإرسال و الإستقبال للرسائل.

Cilk : هي واحدة من لغات البرمجة المتوازية موجهة إلى الأجهزة المتوازية ذات الذاكرة المشتركة. تعتمد على لغة البرمجة C و على أوامر مثل:

الكلمة المفتاح spawn لاعلان التوازي ويوضع في بداية نداء الدالة وكلمة sync للمزامنة.

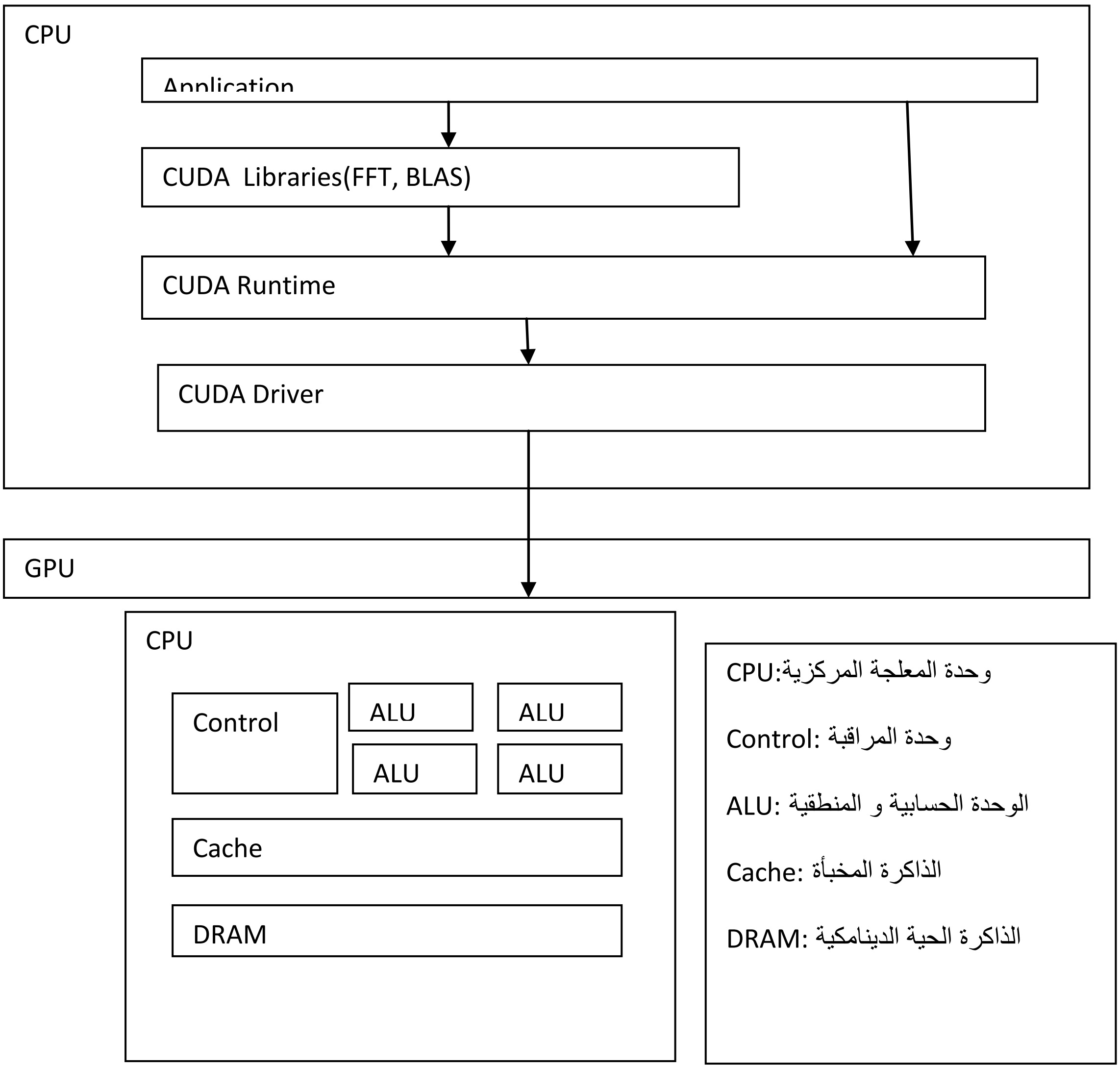
نشير في هدا الصدد إلى أن جميع لغات البرمجة هده لها ارتباط كبير ببنية الحاسوب وخصائصه. ومن أشهر لغات البرمجة المتوازية و التي ظهرت حديثا و التي لا تتطلب الحواسيب الكبيرة أو الانتقال إلى مخابر مراكز البحث كي يمكن استعمالها لغة البرمجة **كودا (CUDA)** و التي يمكن استعمالها في الحواسب الصغيرة والتي تتضمن البطاقة الإلكترونية Nvidia[18].

(CUDA : Compute Unified Device Architecture)

يوضح الشكل (4−26) الموالي بنية كل من المعالج المركزي (CPU) والمعالج الرسومي (GPU) و كيفية التواصل بينهما قصد إجراء الحسابات و سنعرض الطريقة لاحقا.

[18]: Yacine Amara ، مصدر سبق ذكره ، صص.26-27

الشكل(4-26): تطبيق برنامج كودا علىCPU/GPU



GPU									
CONTROL	CACHE	ALU	ALU	ALU	..	..	..	..	ALU
CONTROL	CACHE	ALU	ALU	ALU					ALU
..	..								
DRAM									

المصدر : Amara yacine، مصد سبق ذكره ، ص.26

يظهر لنا هذا الشكل(4−26) بنية حاسوب بسيط يتكون معالج الرسومات فيه من عدة وحدات مراقبة تحتوي كل واحدة من ذاكرة مخبأة ذات سعة صغيرة و ذاكرة حية شاملة و عدد كبير من الوحدات الحسابية و المنطقية والذي يمكن استغلاله في إجراء الحسابات بالتوازي. أما المعالج الرئيسي فيحتوي على وحدة مراقبة وحيدة ، عدد قليل من الوحدات الحسابية و المنطقية ، ذاكرة مخبأة وحيدة و ذاكرة حية شاملة.

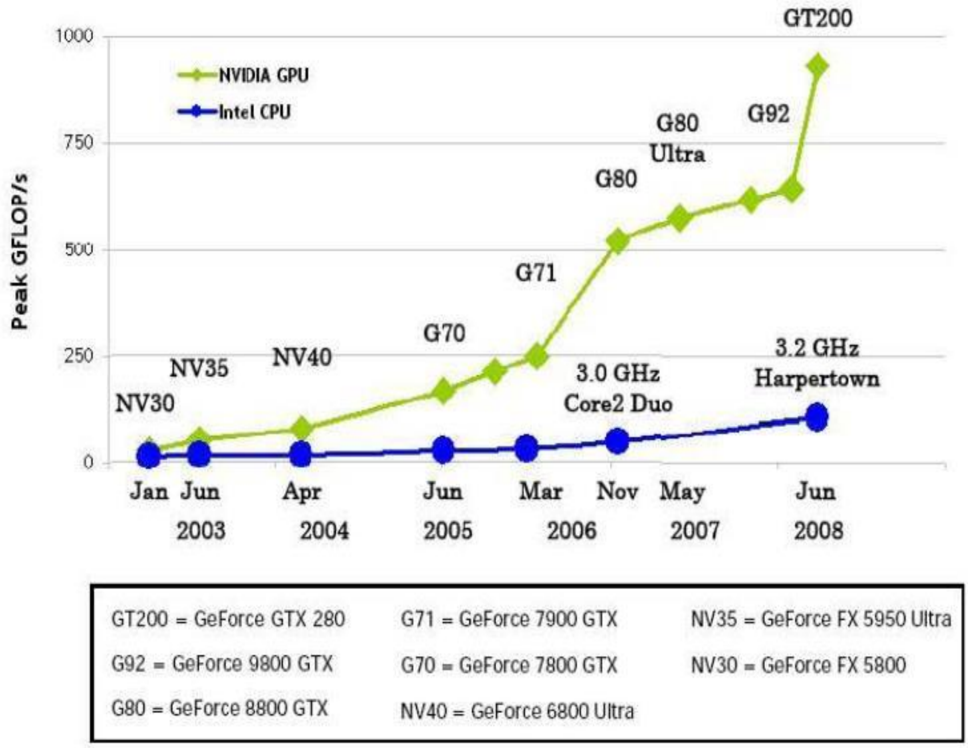
تحتمل بنية كودا عدة لغات معا مثل لغة البرمجة C. C++, Fortran، و يحتوي

على المكتبات BLAS (مكتبة الجبر الموجودة مسبقا) و CuFFT (مكتبة Fourier للتحويل السريع).

تحتمل بنية كودا عدة لغات معا مثل لغة البرمجة C. C++, Fortran، و يحتوي

على المكتبات BLAS (مكتبة الجبر الموجودة مسبقا) و CuFFT (مكتبة Fourier للتحويل السريع).

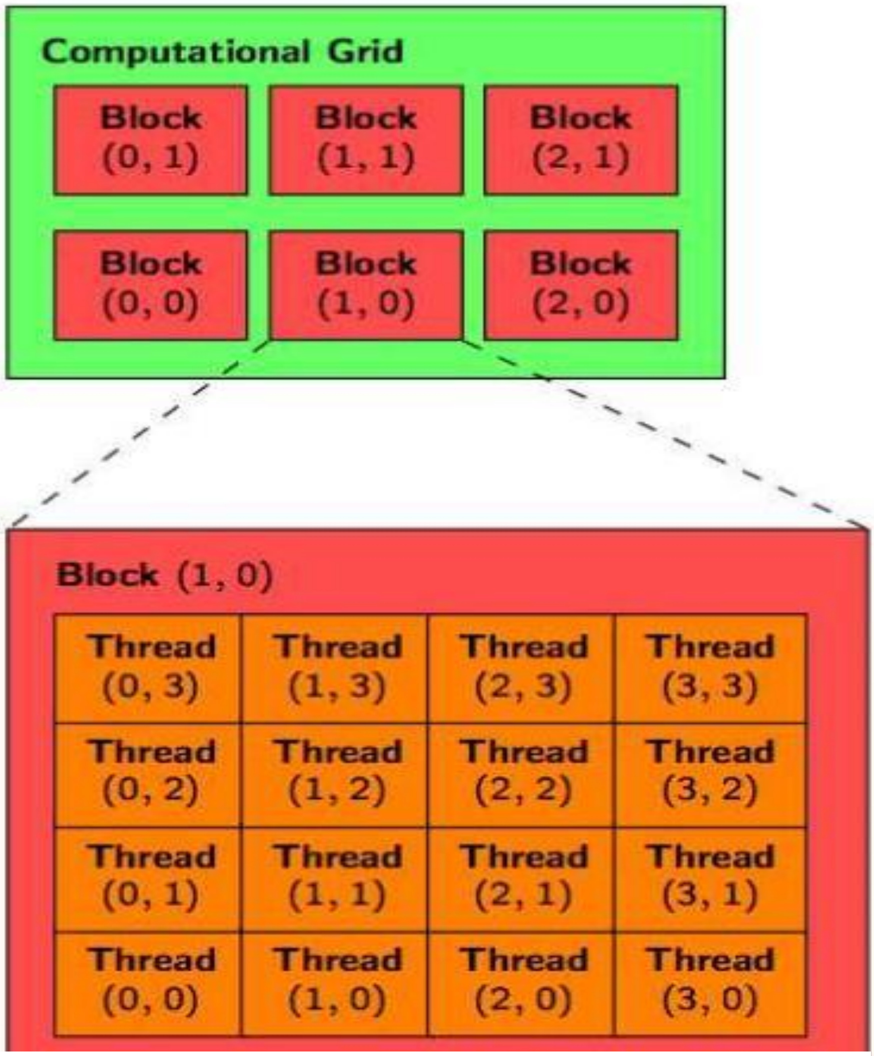
الشكل (4–27):مقارنة التطور عبر الزمن لسرعة التنفيذ بين المعالج الرسومي NVIDIA GPUو المعالج الرئيسي Intel CPU.



المصدر: Eric Goubault, " Introduction à CUDA ", Ecole polytechnique, PARIS- , p.3:
SACLAY,2012

نلاحظ من خلال الشكل(4–28) أن السرعة القسوى (peak GFLOP/s) للمعالج (GPU) تفوق سرعة المعالج الرسومي (GPU) و تزداد مع مرور الزمن.

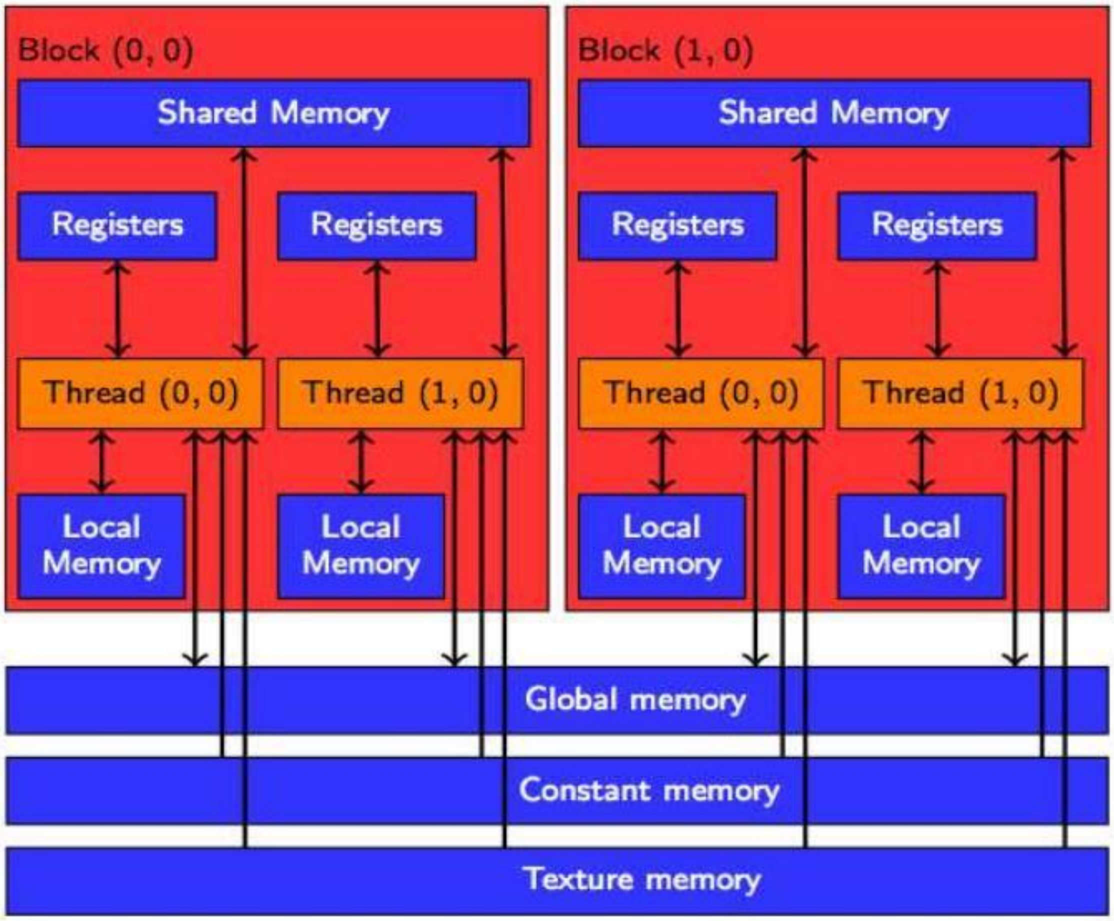
الشكل (4−28).: شكل الشبكة الحسابية (computational grid)



المصدر: Eric Goubault, ، مصدر سبق ذكره ، ص.10

يتضح من خلال الشكل(4−28) أن الشبكة الحسابية للمعالج الرسومي مكونة من عدة كتل(blocks) و أن كل كتلة بدورها مكونة من عدة خيوط التنفيذ(threads).

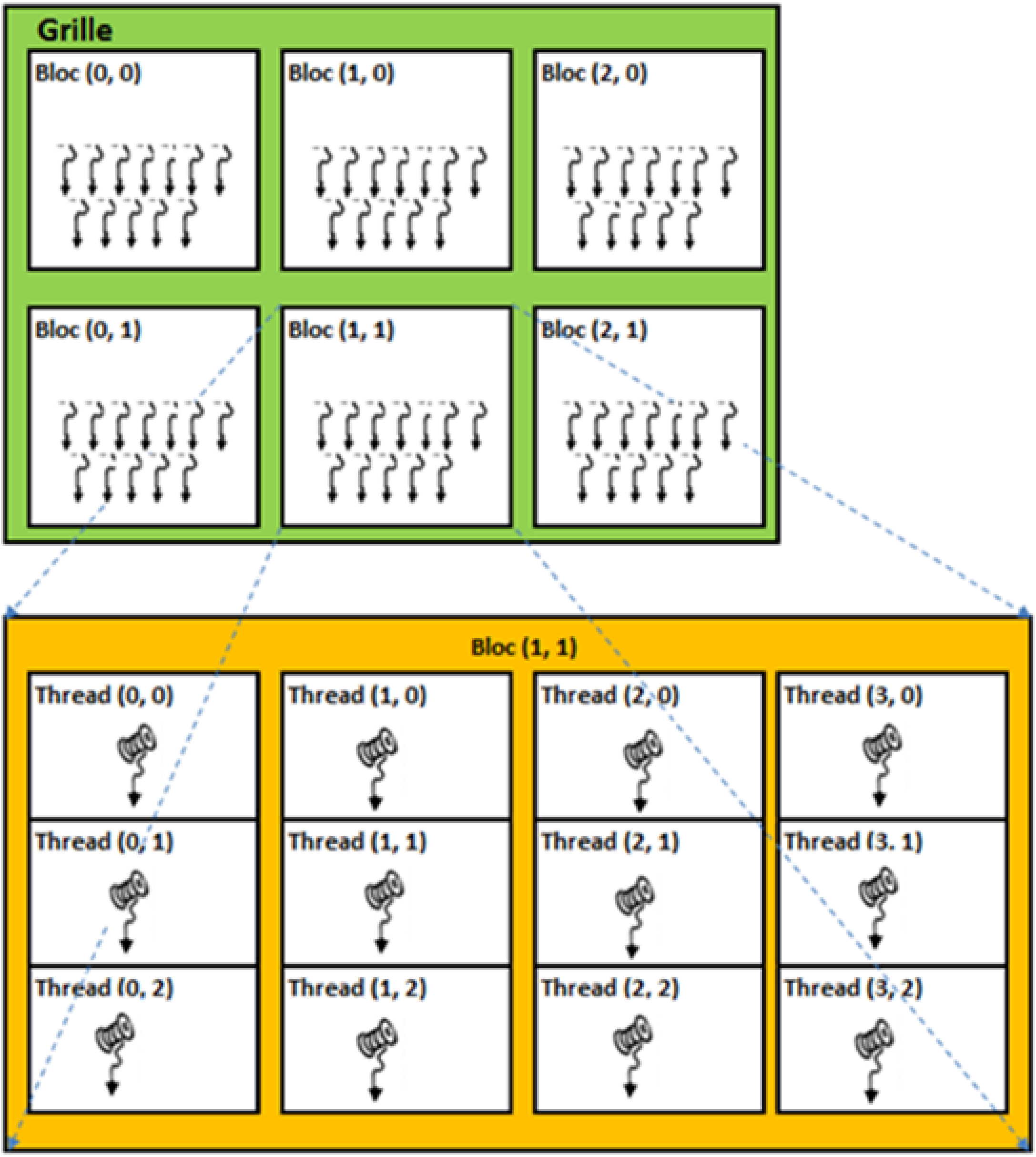
الشكل (4–29).: نموذج بنية ذاكرة المعالج الرسومي (GPU)



المصدر: Eric Goubault ، مصدر سبق ذكره، ص.11

يتضح من الشكل(4–29) أن لكل كتلة (block) ذاكرة المشتركة و بالتالي لا يمكن الولوج إليها (القراءة و الكتابة) إلا من طرف خيوط التنفيذ (threads) المكونة لهذه الكتلة و أن لهذا الأخير أي خيط التنفيذ، ذاكرة محلية (local memory) و سجلات (registers)خاصة به. بينما الذاكرة الشاملة (Global memory) مشتركة للجميع.

الشكل (4-30): تنظيم خيوط التنفيذ (Tread GPU) على بنية كودا(cuda)



المصدر: http://blogs.msdn.com/devpar ، (Introduction à la programmation GPU)،ص3

الشكل(4-30) يوضح تنظيم و توزيع خيوط التنفيذ على شبكة المعالج الرسومي (GPU).

2.6.4 طريقة كتابة برنامج في كودا.

-كتابة رمز البرنامج لوحدة المعالجة المركزية (CPU) .

-تحديد الإجراءات (الدوال)المكلفة .

-التحقق من الطابع (SIMD) للإجراء.

-نسخ البيانات إلى الذاكرة الشاملة للمعالج (GPU).

-إرسال الإجراء (الدالة) للتنفيذ على المعالج (GPU).

-استعادة النتائج من ذاكرة المعالج (GPU).

-تحسين الأداء باستعمال الذاكرة المشتركة

3.6.4 بروتوكول الاتصالات لنداء النواة، بين المعلج (CPU/HOST) و المعالج(GPU/KERNEL)[19].

المرحلة 1: تحديد و تخصيص المعلمات التي يجب أن يتم نسخها في الجانب GPU لتزويدهم بالدالة النواة (kernel).

المرحلة 2: نسخ المعلمات في ذاكرة المعالج GPU.

المرحلة 3: استدعاء الدالة النواة (kernel).

المرحلة 4: تشغيل الخوارزمية حسب عمارة كودا للبطاقة الرسومية GPU.

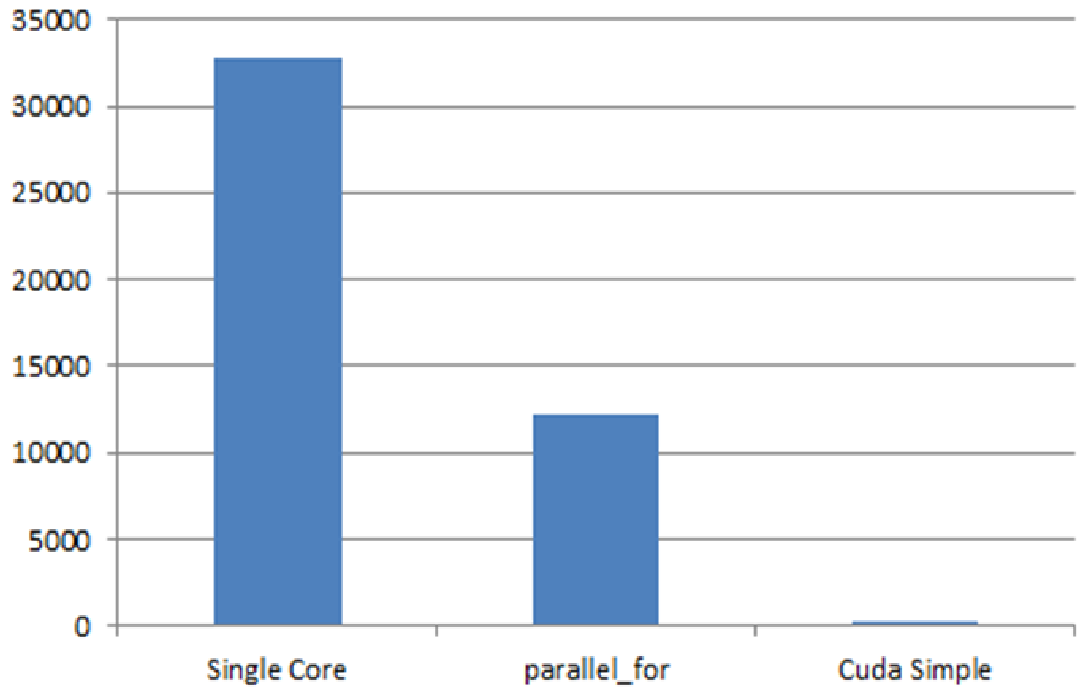
المرحلة 5: العودة من الوضع GPU/KERNEL إلى الوضع CPU/HOST.

المرحلة 6: نسخ النتائج في الوضع CPU/HOST.

المرحلة 7 : تدمير مخازن الذاكرة المستخدمة من قبل الدالة النواة.

[19] :Site http://blogs.msdn.com/devpar,"Introduction à la programmation GPU", p.5

الشكل (4-31): مقارنة حساب المصفوفة المعبر عنه بالمللي ثانية(ms)



المصدر : الموقع http://blogs.msdn.com/devpar **، مصدر سبق ذكره، ص.12**

يتضح من خلال الشكل(4-31) أن زمن تنفيذ برنامج حساب المصفوفة في لغة Cuda أقل بكثير منه في لغة البرمجة المتوازية parallel_for والذي يفوق(10000ms) وأيضا يكاد لا يقارن مع زمن حساب المصفوفة باستعمال معالج وحيد النواة (single core) والذي يفوق (30000ms).

و من أهم النتائج المتوصل إليها من خلال حساب معايير نجاعة الخوارزميات المتمثلة في زمن التنفيذ ، التسريع ، و الفعالية. تبين أن الصيغة المتوازية لهذه الخوارزميات سواء منها خوارزميات الفرز أو خوارزميات نظرية البيان هي الأنجع كونها لها أقل زمن تنفيذ و أفضل تسريع عند تنفيذها مقارنة بصيغها التسلسلية و يظهر هذا خاصة كلما كان حجم معطياتها كبير جدا.

خلاصة:

تطرقنا في الفصل الرابع في البداية إلى تعريف بعض المفاهيم في بحوث العمليات و خاصة منها تلك المتعلقة بنظرية البيان كمفهوم البيان ، المسار، المسار الأقصر و الشجرة بأقل تغطية...ثم بعد ذلك أعطينا نص خوارزمي بريم التسلسلي و المتوازي لإيجاد أدنى شجرة تغطية في البيان.

و بعد ذلك إلى الخوارزميات الخاصة بإيجاد أقصر مسار في البيان نذكر: خوارزمي Dijkstra (من قمة المنبع إلى كل القمم الأخرى في البيان) في صيغته التسلسلية و المتوازية و كذلك خوارزمي Floyd (بين كل زوج من القمم في البيان) التسلسلي و المتوازي.

في المرحلة الثانية، قمنا بتحليل هذه الخوارزميات قصد دراسة مدى جودتها و هذا اعتمادا على حساب بعض المعايير كالتعقيد الزمني(زمن التنفيذ) ، التسريع و فعالية الخوارزمي. بدأنا حساب هذه المعايير على خوارزميات الفرز التسلسلية البسيطة المعروفة ثم مقارنة تعقيدها الزمني. فالخوارزمي ذو التعقيد الزمني الأقل يعد هو الأفضل.

نشير إلى أن خوارزميات الترتيب تدخل في بعض خوارزميات نظرية البيان، نذكر على سبيل المثال خوارزمي Kruskal و الذي يقوم في البداية بترتيب أضلاع البيان تصاعديا قبل انشاء الشجرة بأدنى تغطية.

بعد ذلك، قمنا بحساب المعايير الأخرى(التسريع و الفعالية) على خوارزميات الترتيب المتوازية على غرار الخوارزمي الفردي-الزوجي، خوارزمي الفرز السريع و خوارزمي الفرز بالدمج وهذا باستعمال برنامج (EXCEL) مما سمح لنا بمقارنة فعالية كل خوارزمي في صيغته التسلسلية مع صيغته المتوازية من جهة ومع خوارزميات الترتيب الأخرى من جهة أخرى.

قمنا بعد ذلك بدراسة مدى فعالية خوارزمي Prim و خوارزمي Kruskal في صيغه التسلسلية والمتوازية في خطوة أولى ثم مقارنتها فيما بينها. نفس الشيء قمنا به في خطوة ثانية مع خوارزمي Dijkstra و خوارزمي Floyd. و هذا من خلال حساب زمن التنفيذ ، التسريع و الفعالية.

في مرحلة ثالثة، تطرقنا إلى التعريف بلغات البرمجة المتوازية و مميزاتها. ومن بين هذه اللغات نذكر: OpenMP, Pthread, MPI, Cilk

نشير في هذا الصدد أن جميع هذه اللغات أو المكتبات المتوازية لها ارتباط وطيد ببنية الحاسوب و خصائصه. من أشهر هذه اللغات المتوازية و التي ظهرت حديثا و لا تتطلب حواسيب كبيرة أو مراكز بحث، لغة البرمجة كودا(CUDA) و التي يمكن استعمالها في الحواسيب الصغيرة و التي تتضمن بطاقة الرسومات الإلكترونية (NVIDIA) و التي تستعمل معالج الرسومات GPU في الحسابات.

الخاتمة العامة:

في هذا البحث قمنا بعرض بعض المفاهيم حول المعالجة المتوازية، و توصلنا إلى أن هناك عدة أنواع للبنى المتوازية للحواسب تسمح بتنفيذ البرامج (الخوارزميات) المتوازية. و لكن لكل بنية خاصيتها (ذاكرة مشتركة أو ذاكرة موزعة، عدد المعالجات، نوع شبكة الربط المستعملة).

و كذلك أن البرنامج يمكن نمذجته من خلال مخطط أو بيان (بيان التبعية، بيان السوابق، تدفق المعطيات) و الذي يتضح من خلاله بنية الحاسوب الملائمة لتنفيذ البرنامج عليها.

تبين لنا كذلك أنه ليس كل الخوارزميات قابلة للتوازي وأن مدة تنفيذ الخوارزميات المتوازية أقل بكثير من مدة تنفيذ الخوارزميات المتسلسلة نظريا. و لكن في بعض الحالات يكون العكس هو الأصح. و تكون المعالجة المتسلسلة هي الأفضل. و أن لهذا أسبابه مثلا العدد الكبير للرسائل المتبادلة بين المعالجات والانتظار الحاصل في استقبال هذه الرسائل. و خاصة كما نعلم وأن زمن هذا التواصل بين المعالجات أكبر بكثير من الزمن الذي تتطلبه عملية حسابية من طرف المعالج. أو عندما يكون حجم المعطيات صغيرا، لا يسمح أيضا من ملاحظة الفارق بين زمن التنفيذ في كل من الخوارزمي المتسلسل و الخوارزمي المتوازي.

إن لتوزيع المهام على المعالجات كذلك له تأثير في سرعة التنفيذ البرنامج. الحالة المثلى هي إشغال هذه المعالجات على الدوام حتى لا يبق هناك معالج في الانتظار و بالتالي ضياع لوقت التنفيذ. و لحساب تكلفة أو مدة التنفيذ الخوارزميات المتوازية نماذج تعتمد على تحليل ما يسمى بالتعقيد الزمني للخوارزمي ومقارنته بالتعقيد الزمني للخوارزمي المتسلسل ومن ثم الحكم على من يكون الأنجع.

بالإضافة إلى هذا يمكن تلخيص أهم النتائج كما يلي:

النتائج:

- **بالنسبة لخوارزميات الفرز:**

لاحظنا انخفاض كبير في زمن تنفيذ خوارزمي **الفرز الفردي الزوجي** المتوازي و بنسبة حوالي 50% عن زمن تنفيذ الخوارزمي في صيغته التسلسلية ويتضح هذا خاصة كلما أرتفع حجم المعطيات المدخلة. أما عن التسريع و الفعالية تكون أفضل عندما يكون عدد المعالجات إثنان ($p=2$). و عن زمن تنفيذ

خوارزمي الفرز السريع المتوازي فيقل كذلك بحوالي 50% عن زمن تنفيذه في صيغته التسلسلية و هذا عندما يكون عدد معالجات الحاسوب إثنان و إلى 25% عندما يكون عددها أربعة. و فيما يخص المقارنة بين خوارزمي الفرز الفردي–الزوجي و خوارزمي الفرز السريع، فإن هذا الأخير له أقل تعقيد زمني متوازي Tp و بالتالي زمن تنفيذ سواء تم تنفيذه على حاسوب من معالجين أو أربع معالجات.

- **بالنسبة لخوارزميات نظرية البيان:**

نلاحظ أن التعقيد الزمني المتوازي **لخوارزمي Prim** يقل ب حوالي 50% عن نظيره في الصيغة التسلسلية . وبالتالي فالصيغة المتوازية للخوارزمي هي الأسرع تنفيذا. أما عن رفع عدد المعالجات من إثنان إلى أربع فهو غير فعال. وفيما يخص **خوارزمي Kruskal** فصيغته المتوازية غير فعالة كون تعقيده الزمني المتسلسل Ts أقل بكثير من تعقيده المتوازي Tp و هو ما يظهره حساب التسريع إذ أنه في تناقص كلما أرتفع حجم المعطيات المدخلة.

بالنسبة **لخوارزمي Dijkstra**، فإن تعقيده الزمني التسلسلي Ts أقل بكثير عن تعقيده المتوازي Tp و أن تسريع الخوارزمي المتوازي في تناقص كلما أرتفع حجم المعطيات و بالتالي فالصيغة المتوازية للخوارزمي ليست فعالة و هو ما يفسر أنه ليس دائما يكون الخوارزمي المتوازي له أقل زمن تنفيذ. أما **خوارزمي Floyd**، أظهرت الدراسة أن تعقيده المتسلسل Ts أكبر من تعقيده المتوازي Tp و أن فعالية الخوارزمي المتوازي متساوية سواء تم تنفيذه على حاسوب من معالجين أو من أربعة معالجات.

الصعوبات:

- إن دراسة الخوارزميات المتوازية هذا الفرع الجديد من فروع علم المعلوماتية مسألة صعبة في وقتنا الحالي وخاصة لعدم توفر أجهزة ملائمة في بلادنا تسمح بترجمة هذه الخوارزميات إلى برامج عملية تمكننا من التحقق من النتائج و تحليلها بصفة دقيقة.
- واجهنا كذلك صعوبات في ايجاد مراجع باللغة العربية و التي تناولت موضوع المعالجة المتوازية في الحاسوب.

في الأخير، بالرغم من النتائج التي توصلت لها الدراسة وهناك حاجة إلى المزيد من البحوث في هذا المجال قصد تطويره مستقبلا.

إلا أن الدراسة تعتبر محاولة وتمثل سبيلا لفتح المجال أمام البحوث و المواضيع التي يمكن إقتراحها كتطوير و إثراء هذا النوع من المواضيع الهامة والجديدة في الوقتنا الحالي لمواكبة التطور التكنولوجي من بينها:

المقترحات:

- إجراء دراسات موسعة لكافة المسائل بالأسلوب المتوازي و فهم الطرق التعامل مع الأجهزة المتوازية وشبكات اتصالها لتحديد الخوارزميات المناسبة لهذه المسائل.
- تطبيق الخوارزميات المتوازية في ميدان بحوث العمليات بشكل أوسع و مقارنتها بالخوارزميات المتسلسلة و خاصة منها تلك المصنفة بالصعبة كخوارزمي التاجر المتجول وخوارزمي حقيبة الظهر و غيرها.

قائمة المراجع و المصادر :

I- المراجع باللغة العربية:

1- فهد بن صديق، " **أساسيات الشبكات**"، مراكز نيو هور إزن، جازان.

2- عماد فتاش، " **الحاسبات المتوازية و الخوارزميات المتوازية**"، جامعة الملك سعود، كلية العلوم، قسم الحاسب الآلي، 2002.

3- ميساء بدر حمشو ، " **الخوارزميات المتوازية و الخوارزميات الموزعة و تطبيقاتها في بحوث العمليات**"، جامعة دمشق، 1998 .

4- رند عمران مصطفى الأسطل، "**بحوث العمليات و الأساليب الكمية في صنع القرارات الإدارية**"، فلسطين، الطبعة السادسة 2016.

5- كندة زين العابدين، " **خوارزميات المعالجة المتوازية و برمجتها**"، جامعة دمشق، 2006.

6- محمد وجيد النما، "**الطرق التكرارية المتوازية**"، جامعة الموصل، العراق، 2009.

7- محمد واجد محمد علي ، "**الطرائق التكرارية المتوازية**" ، مجلة كلية التربية للبنات، جامعة الموصل، العراق، 2009.

- نغم ثروت سعيد، " **الإخفاء باعتماد الخوارزميات المتوازية** " ، قسم علوم الحاسب / كلية التربية ، مجلة التربية والعلوم المجلد (25) ، العدد (1) ، جامعة الموصل ، 2011.

II- المراجع باللغة الأجنبية:

II-**1 الكتب الورقية و الإلكترونية**

8- **"Introduction to parallel Algorithms and Parallel Program design"**, University of Oregon , pp. 2-17. http://ipcc.cs.uoregon.edu/lectures/lecture-12-algorithms.pdf

9- Alain Billionnet, **"Mathématique-Recherche opérationnelle"**, 2010-2011.

10- Alain Cazes&Joelle Delacroix, **"Architecture des machines et des systemes informatiques"**, 5ème edition Dunod, France.

11- Arnaud Labourel, "**Programmation parallèle et distribuée"**, Université de Provence, France ,2012.

12- Aurélia Marchand, "**Les architectures parallèles**-MPI",pp.2-8.

13- Benoit Semelin, " **Programmation parallèle pour le calcul scientifique**", France, 2014. http://aramis.obspm.fr/~semelin/cours_parallelisme.pdf

14- Bernard Petit, **"Architecture des réseaux"**, 4ème édition ellipses, BU Paris Dauphine, 2012.

15- Boualem Benmaazouz, "**Recherche opérationnelle de gestion**", ATLAS EDITIONS, 1995.

16- Brahim Bekhti, **"l'essentiel de la mico-informatique",** Edition N°1 ISP Ouargla, 1999.

17- Brice Goglin, **"Systèmes parallèles et distribués"**, INRIA Bordeaux, 2011-2012

18- C. Jacquemard, G. Laurent, **"Mémento de C et C++ "**, (support de cours), 2011.

http://www.glaurent.free.fr/cours/MementoC++.pdf

19- Christian Levault, **"Evaluation des Algorithmes Distribuées"**, édition Hermès, Paris, 1995.

20- Christophe Duhamel, **"Outils d'Aide à la Decision"**, 2014, pp. 19-40.

21- Christophe Haro, **"Algorithmique: Raisonner Pour Concevoir"**, Edition ENI, France, 2009.

22- Christophe Pera, **"Introduction à MPI :Message Passing Message"**, 2012/2013.

http://lyoncalcul.univ-lyon1.fr/ed/DOCS_2012-2013/mpi.pdf

23- Cosnard Michel&Trystram Denis, **"Algorithmes et architectures parallèles"**, Paris inder éditions, 1993.

24- Cyril Gavoile, "**Algorithmes distribués** ", Université de Bordeaux1, 2005.

25- Daniel C.Hyde.,**" Introduction to the Principles of Parallel Computation**" Bucknell University, 1998,http://www.eg.bucknell.edu/~cs366/textbook-pdf/parallel98.pdf

26- Daniel Dromard&Dominique Seret, **"Architecture des réseaux "**, Pearson éducation, France,2006.

27- Daniel Etiemble, **" Algorithmes de tri parallèles"**, Paris, 2003.

28- Document, **"Evaluation et critères de performances d'un calcul parallèle".**

https://repo.zenk-security.com/Others/Evaluation%20et%20criteres%20de%20performances%20d.un%2

29- Dominique De Werra&Thomas M.Liebling, **"Recherche opérationnelle pour Ingénieurs"**, edition Presses polytechniques et universitaires romandes, 2009.

30- Dongwook Lee, "**Hardware and Software Implementation of Prim's Algorithm for efficient Minimum Spanning Tree computation"**, University of Texas, Austin, USA .

31- Emmanuel Caillaud, **"Recherche Opérationnelle pour le Génie Industriel"**, 2014-2015.

32- Emmanuel Lazard&Pièrre Mounier_Kuhn, **"Histoire Illustrée de l'informatique"**, BU Paris Dauphine, Paris.

33- Eric Goubault & Sylvie Putot, "**Calcul parallèle et distribué** ", Ecole polytechnique, PARIS-SACLAY, 2014.

34- Eric Goubault, " **Introduction à CUDA** ", Ecole polytechnique, PARIS-SACLAY,2012.

35- Eric Trichet, **"Introduction à la complexité algorithmique "**, Université de Limoges.

36- Fabian Bastin, **"Modèles de recherche opérationnelle"**, Université de Montréal, 2010.

37- Fayez Gebali, "**Algorithms and parallel computing**", University of Victoria, Canada.

38- Fernando Silva, **"Parallel Algorithms Sorting",** pp.1-15.

39- Frédéric Vivien, "**Algorithmique avancée",** IUP 2, 2002.

40- George Karypis, "**Introduction to parallel computing:Graph Algorithms**", pp. 2-14.

41- Gergel V.P , **"Introduction to Parallel Programming: Parallel Methods for Sorting"**, Nizhny Novgorod, 2005, pp.5-30.

42- Gergel V.P , "**Parallel Methods for Graph Calculations",** Nizhny Novgorod, pp.2-10.

43- Grama et. al. , "**Parallel Graph Algorithm"**, 1994.

44- Guillermo Andrade B**.," Introduction à la programmation parallèle multi-cœurs",INRIA Rennes, 2010.**

45- Guy Tremblay, **"Programmation concurrente et parallèle**", 2016.

46- Hugues Leroy, **" Parallélisme et programmation par échange de message Utilisation de la bibliothèque MPI"**, INRIA Alger, 2008.

47- Ian FOSTER, **"Designing and building parallel programs"**, Addison-Wesley, 1995.

48- Irène Guessarian, **" Quelques Algorithmes Simples de Tris"**, 2012.

https://www.irif.fr/~ig/coursalgo.pdf

49- Jacques Jorda&Abdelaziz Mzoughi, **"Manuel: Architecture de l'ordinateur"**, Dunod , Paris, 2012

50- Jean Fruitet, **"Introduction à l'informatique et programmation en langage C"**, I.U.T. de Marne.La-Vallée, 1999.

51- Jean RENAUD, **"Planification Opérationnelle des Projets"**, INPL-ENSGSI, 2000.

52- Jun Zhang, **"Parallel computing: Performance and scalability"**, University of Kentuckey.

53- Lavallée Ivan, **"Algorithmique parallèle et distribuée"**, Paris Hermès 1990.

54- Loic Gouarin, **"Introduction au calcul parallèle**", CNRS, 2012.

55- Loic Helouet, " **Algorithmes et Graphes** ", Eyrolles , 1985.

56- M. Amara Yacine, "**Le GPU : un moyen pour le calcul intensif**", Ecole Militaire polytechnique, Alger.

57- M. Dalmau, **"Les superordinateurs",** IUT de BAYONN, 2010.

58- M. Eleuldj, **"Architectures parallèles "**, Département Génie Informatique, EMI, septembre 2014.

59- Mattieu Pérotin, **"Synthèse des outils de parallelisation",** LI Tours.

60- Mikhail J. Atallah, Marina Blanton**, "Algorithms and theory of computation Handbook",**CRC Press Taylor & Francis Group, New York.

61- Mohsine Eleuldj, **"Systèmes et Traitement parallèles"**, EMI, 2014.

62- N. Hameurlain, **"Architectures Parallèles"**, Université de PAU ,(http://www.univ-pau.fr/~hameur)

63- N. SBIHI, **"Polycopié de Recherche Opérationnelle"**, Ecole Mohammadia d'Ingénieurs, Maroc.(www.almohandiss.com)

64- Olivier Leroy, **"L'anglais de l'informatique"**, 2007.

65- Patrick Deiber, " **Les processus légers(threads)"**, Probatoire CNAM Versailles, 1999.

66- Paul Feautrier, "**Méthode élémentaire de parallèlisation"**, ENS de Lyon, novembre 2008.

67- Philipe Marquet, " **Introduction à la programmation parallèle** ", Université des sciences et technologie de Lille, France.

68- Philippe Marquet, **"Programmation parallèle et distribuée"**, Université des sciences et technologie Lille, 2010.

69- Piere_Alain Goupile, **"Technologie des ordinateurs et des réseaux"**, 9ème édition Dunod, 2005.

70- Pierre Delisle, **"Introduction au parallélisme Architectures Algorithmique Programmation"**, Université de Reims, 2016.

71- Pierre_Fraigniaud, **"Algorithme parallele et distribuée"**, Univ. Paris Diderot, 2017.

72- R. Dumont, **"Algorithme P2: La complexité "**, 2009, pp. 2-25.

73- Robert Faure, "**Précis de recherche opérationnelle"**, Dunod, 1985.

74- Rodolphe Buda, **"Les algorithmes de la modélisation: une analyse critique pour la modélisation économique"**, Université de Paris 10, 2001.

75- Ronan Keryell, **"Architecture des machines parallèles modernes"**, ENST Bretagne, 2006.

76- Sabine De Bliek, "**7 défis pour découvrir la théorie des graphes"**, Institut Saint Laurent, 2010.

77- SALGUES Floriane, **"La recherche opérationnelle est un outil d'aide à la décision pour le marketing"**,29/10/2015,http://www.e-marketing.fr/Thematique/data-1091/Breves/recherche-opérationnelle-est-outils-aide-decision-marketing-260807.htm#8RJmF75MthGSifxQ.97

78- Souad EL Bernousi ,"Recherche Opérationnelle", Rabat(www.fsr.ac.ma/ANO/)

79- Stéphane Grandcolas, **" Algorithmes de tris", p.3.**

80- Vincent Bouchite, **"Graphes et Algorithmique des graphes"**, (support de cours), Ecole Normale Supérieure de Lyon, 1998.

81- Violaine Louvet, **"Architecture des ordinateurs concepts du parallélisme"**, école doctorale Mathlf, 2010.

82- Vivek Sarkar, **"Parallel Graph Algorithms"**, Rice University, 2008.

83- Yves Nobert&Roch Ouellet, **"Méthodes d'optimisation pour la gestion"**, 2ème édition chenelière education,2016.

84- Yves Robert & Arnaud Legrand, **"Algorithmique parallèle"**, Dunod Paris, 2003.

(https://mpra.ub.uni-muenchen.de/3926/)

85-Thomas Cormen , Charles Leiserson, Ronald Rivest , Clifford Stein , **"Introduction à L' algorithmique: Cours et exercices"**, Dunod, Paris, 2004.

86-Vincent Boyer, Didier El Baz, " **Résolution parallèles des problèmes d’optimisation en variables 0-1"**, LAAS-CNRS, (2009).

87-Yvon G. Perreault, "**Recherche opérationnelle Techniques Décisionnelles** ", Gaetan morin, 1980.

II-2 المجلات و المداخلات:

88- Armelle Merlin, **"Parallélisme Algorithmes PRAM"**, Laboratoire d'Informatique Fondamentale d'Orléans(LIFO).

89- Bernard Roy, **"Regard historique sur la place de la recherche opérationnelle et de l'aide à la décision en France",** 2006(http://msh.revues.org/3570)

90- Khaled Thabit and Afnan Bawazir**, "A Novel Approach of Selection Sort Algorithm with Parallel Computing and Dynamic Programming Concepts",** *Computer Science Department, Faculty of Computing and Information Technology, King Abdulaziz University, JKAU: Comp. IT.,* **Vol. 2**, pp: 27 - 44 (2013 A.D./ 1435 A.H.)DOI: 10.4197 / Comp. 2-2.

91- Maha Saada &Huda Saada&Mohammad Qatawneh, **"Performance Evaluation of Parallel Sorting Algorithms"**, International of Advanced Science and Technology, Vol.95(2016), pp. 2-14.

92- Philipe Chrétienne, **"La recherche opérationnelle: La science pour 'mieux comprendre' et 'mieux résoudre' les problèmes décisionnels"**,(Séminaire), ISIMA CLERMONT-FERRAND, 2005، 26-24

93- Piere Rousseau, **"Présentation de CUDA"**, (Séminaire), Université de Limoges, 2007.

94- Verhulst Michel, **"La recherche opérationnelle et la gestion économique des entreprise**s", In revue économique, 1954, pp.604-612.http://www.persee.fr/doc/reco_0035-2764_1954_num_5_4_407061

95- Vipin Kumar, Ananth Grama, Anshul Gupta, and George Karypis, **"Introduction to parallel computing",** Redwood City :Benjamin/Cummings, 1994

96- Wei Wang," **Design and implementation of GPU – Based Prim's algorithm"**, I. J. Modern Education and Computer science, 2011

97- Wood, Sohi, Sorin, Roth, Hill, **, "Multiprocessing and Multitreading**", ECE252/CPS220, . Lecture Notes,2005.

98- Zaineb T. Baqer, " **Parallel computing for sorting Algorithms**", Baghdad science journal, vol. 11(2), 2014.

II-3 الأطروحات:

99- Abdelamine Boukedjar, **"Le calcul multi GPU et optimisation combinatoire "**, rapport de stage, Université de Toulouse, 2010/2011.

100- Alexandre Borghi, "**Adaptation de l'algorithmique aux architectures parallèles**" , Doctorat soutenu, Université Paris Sud XI, 2011

101- Cedric CHEVALIER, "**Conception et mise en œuvre d'outils efficaces pour le partitionnement et la distribution parallèle de problèmes numériques de très grande taille",** Université Bordeaux1, 2007.

102- Daouda Traoré, "**Algorithmes parallèles auto-adaptatifs et application**", Institut polytechnique de Grenoble, 2008,pp.20-28.

103- Denis Trystram, " **Quelques résultats de complexité en algorithmiques parallèle et systolique", (**doctorat informatique), , INPG, Grenoble, 1988.

104- Laurent Viennot, **"Quelques algorithmes parallèles et séquentiels de traitement des graphes et applications",** Université Paris-Diderot, Paris 7, 1996.

105- Pierre Delisle, **"Parallélisation d'un algorithme d'optimisation par colonies de Fourmis pour la résolution d'un problème d'ordonnancement industriel"**, 2002.

106- Rédha LOUCIF, **" Parallélisation d'Algorithmes d'Optimisation Comdinatoire",** Faculté des sciences, Université HADJ LAKHDAR, BATNA, 2014.

107-Xavier Meyer, "**Etude et implémentation de l'algorithme du simplexe standard sur GPU**", Université de Genève, 2011.

108- Yaroub ELLOUMI, **"Parallélisme des nids de boucles pour l'optimisation du temps d'exécution et de la taille du code"**, doctorat soutenu, Université Paris-Est , 2013.

II-4 المواقع الإلكتونية:

109**- classement des superordinateurs TOP-500**, www.top500.org

110-Karim Baina,"**Programmation avancée**",ENSIAS-Rabat(Maroc), https://www.youtube.com/watch?v=X37E1wAT5Wg

111- **systèmes partagés et multiprogrammation**, www.wikibooks.org

الملاحق

الملحق1: برامج حاسوبية

1- أمثلة في لغة البرمجة C

أ- برنامج Prim لإيجاد أدنى شجرة تغطية(MST).

```
//Programme de Prim pour trouver l'arbre de recouvrement minimale dans un
//graphe.
#include <stdio.h>
#include <stdlib.h>
#define TAILLE 6                    /* nombre de sommets du graphe */
#define MAXINT 1000                        /* un tres grand entier */

typedef struct cellule                     /* noeud, poids et pointeur */
{       int numero;
        int poids;
        struct cellule *suivant;
} Cellule, *LISTE;

LISTE graphe[TAILLE];                      /* graphe = tableau de listes */
int D[TAILLE];                             /* Les distances trouvees a chaque
instant */
int queue[TAILLE];                  /* file pour les sommets selon D croissant */
int ordre[TAILLE];                  /* Ordre de chaque sommet dans la file selon
D croissant */
int explore[TAILLE];                /* Pour les sommets deja ajoutes */
int C[TAILLE];                      /* pour le predecesseur de chaque sommet
ajoute */
int u;

/* Alloue un noeud sans initialiser */
```

```
Cellule *AlloueCellule()
{
  Cellule *cell = (Cellule *) malloc(sizeof(Cellule));
  return cell;
}

/* Cree un noeud en l'initialisant avec deux valeurs. */
Cellule *CreeCellule(int cle, int lon)
{
  Cellule *n = AlloueCellule();
  n->numero = cle;
  n->poids = lon;
  n->suivant = NULL;
  return n;
}

/* Initialisation des diverses variables */
void initialisation(void)
{
  int i;
  for(i = 0; i<TAILLE; i++)
  {
    D[i] = MAXINT; queue[i] = i; ordre[i] = i; explore[i] = 0;
  }
  D[0] = 0; C[0] = -1;                    /* On commence par 0 */
}

/* Operation sur les files d'attente */
void echanger(int i, int j)          /* echanger deux sommets dans la queue */
{
```

```
  int a;
  ordre[queue[i]] = j; ordre[queue[j]] = i;
  a = queue[i]; queue[i] = queue[j]; queue[j] = a;
}

/* retablit la condition de file de maniere descendante sur les longueur positions de la file */
void echange_descendant(int longueur, int i)
{
  int fils=2*i+1;
  if(fils>=longueur) return;
  if((fils<longueur-1)&&(D[queue[fils]]>D[queue[fils+1]])) fils=fils+1;
  if(D[queue[i]]>D[queue[fils]])
   {
      echanger(i,fils);
      echange_descendant(longueur, fils);
   }
}

/* retablit la condition de file de maniere ascendante sur les longueur positions de la file */
void echange_montant(int longueur, int i)
{
  int pere=(i-1)/2;
  if(D[queue[i]]<D[queue[pere]])
   {
      echanger(pere,i);
      echange_montant(longueur, pere);
   }
}
```

```
/* retourne le minimum dans la file et retablit l'ordre */
int minimum(int longueur)
{
  int a = queue[0];
  explore[a] = 1;
  if(longueur > 1)
  {
    echanger(0, longueur-1);
    echange_descendant(longueur-1, 0);
  }
  return a;
}

/* algorithme de Prim. Premiere etape, on ajoute un sommet, puis on actualise
*/
void prim_iteration(int longueur)        /* nombre de sommets non encore inclus
*/
{
  Cellule *c;
  int a;
  a = minimum(longueur);                 /* extraction du minimum */
  int j;
  for(c=graphe[a]; c != NULL; c=c->suivant)
  {
    j = c->numero;
    if((explore[j] == 0)&&(c->poids < D[j]))
    {
      D[j] = c->poids; C[j] = a; echange_montant(longueur-1, ordre[j]); /* si
actualisation */
```

```
    }
  }
}

void prim(void)
{
  int i;
  initialisation();
  for(i=TAILLE; i>0; i--) prim_iteration(i);
}

/* Symetrise le graphe */
void symetrise(void)
{
  int i, j;
  Cellule *c, *u;
  for(i = TAILLE - 1; i >= 0; i--)        /* Evite de parcourir les cellules ajoutees
*/
  for(c = graphe[i]; c != NULL; c = c->suivant) /* Parcourt les successeurs j de
i */
  {
    j = c->numero; u = CreeCellule(i, c->poids);
    u->suivant = graphe[j]; graphe[j] = u;     /* Ajoute i comme successeur de j
*/
  }
}

int main(void)
{
  int i; Cellule *c;
```

```
  int poids_total = 0;
  Cellule *c1, *c2, *c3;
  c1 = CreeCellule(1,6); c2 = CreeCellule(2,1); c3 = CreeCellule(3,5);
  graphe[0]=c1; c1->suivant=c2; c2->suivant=c3;
  c1 = CreeCellule(2,5); c2 = CreeCellule(4,3);
  graphe[1]=c1; c1->suivant=c2;
  c1 = CreeCellule(3,5); c2 = CreeCellule(4,6); c3 = CreeCellule(5,4);
  graphe[2]=c1; c1->suivant=c2; c2->suivant=c3;
  c1 = CreeCellule(5,2);
  graphe[3]=c1;
  c1 = CreeCellule(5,6);
  graphe[4]=c1;
  graphe[5]=NULL;
  symetrise();
  prim();
  for(i=0; i<TAILLE; i++)
  {
    poids_total = poids_total + D[i];
    printf("arete %d %d, poids %d\n", i, C[i], D[i]);
  }
  printf("poids total %d\n", poids_total);
}
```

ب- برنامج جمع مصفوفتين (c = a + b)

```
/* calcul  de la somme de deux matrices c=a+b */
float *c; /* resultat */

void  add_matrix(float  *a,  float  *b,  int  N) {
      for  (int  i=0;  i<N;  i++)
            for (int  j=0;  j<N;  j++)
                  c[i+j*N]=a[i+j*N] + b[i+j*N]; }
```

```
int main(int argc, char **argv) {
    float *x, *y;
    int N=16;
    x= (float *) malloc(N*sizeof(float));
    y= (float *) malloc(N*sizeof(float));
    c= (float *) malloc(N*sizeof(float));
    (.......)
    add_matrix(a, b, N); }
```

ج- برنامج جمع شعاعين c= (a + b)

```
// Programme en C pour calculer la somme de deux vecteur a et b.
//

#include "stdafx.h"
#define N 10
void add(int *a, int *b, int *c) {
        int tid =0; // le premier indice est zero
        while (tid < N){
                c[tid] = a[tid] +b[tid];
                tid += 1;  // il n'y a qu'un seul CPU, donc on progresse de 1
en 1
        }
}

int _tmain(int argc, _TCHAR* argv[])
{
        int a[N], b[N], c[N];
        // on remplit les tableaux a et b sur le CPU
        for (int i=0; i<N; i++) {
```

```
        a[i] = -i;
        b[i] = i*i;
    }

    add( a, b, c) ; // appel de la fonction
    // Affichage du resultat
    for (int i=0; i<N; i++) {
        printf("%d + %d = %d\n", a[i], b[i], c[i] );
    }
    return 0;
}
```

2- أمثلة في لغة البرمجة المتوازية كودا(CUDA)

أ- جمع المصفوفات (a+b = c)

```
 /* programme de somme de matrice en cuda */

const int N=1024;

const int blocksize = 16;

__global__ void  add_matrix(float *a, float *b, float *c, int N)

{

   int i = blockIdx.x * blockDim.x + threadIdx.x ;

   int j = blockIdx.y * blockDim.y + threadIdx.y ;

   int index = i + j*N;

   if( i < N  &&  j < N)
```

```
                c[index] = a[index] + b[index];

}

/*programme principal  */

int main() {

   float *a  =  new  float[N*N];

   float *b  =  new  float[N*N];

   float *c  =  new  float[N*N];

   for (int i = 0; i < N*N; ++i  ) {

                  a[i]  =  1.0f; b[i]  = 3.5f; }

  float  *ad,  *bd,  *cd;

  const  int  size  =  N*N*sizeof(float);

  cudaMalloc((void**)&ad, size );

  cudaMalloc((void**)&bd, size );

cudaMalloc((void**)&cd, size );

cudaMemcopy(ad, a, size,  cudaMemcpyHostToDevice);

cudaMemcopy(bd, b, size,  cudaMemcpyHostToDevice);

dim3  dimBlock(blocksize,  blocksize) ;

dim3  dimGrid( N/dimBlock.x,  N/dimBlock.y) ;

add_matrix<<<dimGrid, dimBlock>>>(ad, bd, cd, N ) ;
```

```
    cudaMemcpy(c, cd, size,  cudaMemcpyDeviceToHost ) ;
    cudaFree(ad ) ;
    cudaFree(bd ) ;
    cudaFree(cd ) ;
    delete[]  a ;
    delete[]  b ;
    delete[]  c ;
    return EXIT_SUCCESS;
}
```

ب- برنامج لجمع شعاعين a و b و النتيجة في الشعاع .c

```
// Programme de somme de deux vecteurs a et b.
#define N  10
int main(void) {
        int a[N], b[N], c[N];
        int *dev_a, *dev_b, *def_c;
        // allocation mémoire sur le GPU
        HANDLE_ERROR ( cudaMalloc ( (void**)&dev_a, N*sizeof(int) ) );
        HANDLE_ERROR ( cudaMalloc ( (void**)&dev_b, N*sizeof(int) ) );
    HANDLE_ERROR ( cudaMalloc ( (void**)&dev_c, N*sizeof(int) ) );
        //remplissage des tableaux a et b sur CPU
        for (int i=0; i<N; i++) {
                a[i] = -i;
                b[i] = i*i;
        }
```

```
// copie des tableaux a et b sur le GPU
        HANDLE_ERROR ( cudaMemcpy ( dev_a, a, N*sizeof(int),
cudaMemcpyHostToDevice) );
        HANDLE_ERROR ( cudaMemcpy (dev_b, b,  N*sizeof(int),
cudaMemcpyHostToDevice) );
        add<<<N,1>>>(dev_a, dev_b, dev_c);
        //copie du tableau c du GPU vers le CPU
        HANDLE_ERROR ( cudaMemcpy (c,dev_c, N*sizeof(int),
cudaMemcpyDeviceToHost) );
        //Affichage du resultat
        for (int i=0; i<N; i++) {
            printf( "%d + %d = %d\n" , a[i], b[i], c[i] );
        }
        //liberation de la mémoire allouée au GPU
        cudaFree( dev_a)
        cudaFree( dev_b)
        cudaFree( dev_c)
```

الملحق2 : جدول المصطلحات:

عربي	فرنسي	إنجليزي
التسريع	accélération	speedup
الجامع	additionneur	Adder
الخوارزمي المتوازي	Algorithme parallèle	Parallel algorithm
الخوارزمي المتسلسل	Algorithme séquentiel	Sequential algorithm
رقمي ، تماثلي	Analogique/numérique	Analog/digtal
التجميع	Assembleur	Assembler
طريق المعطيات	Chemin de donnée	Datapath
المبدلات	Commutateur	Switches
المترجم	Compilateur	Compiler
فرق-تسد	Diviser pour régner	Divide and conquer
التعليمة	Instruction	Instruction
التكامل على نطاق واسع	Intégration à grande échelle	Large scale integration
العتاد	Matériel	Hardware
مصفوفة المبدلات	Matrice de commutateurs	Matrix swiches
الذاكرة المشتركة	Mémoire partagée	Shared memory
البرمجة الدقيقة	Microprogrammation	Microprograming
متعدد البرامج	Multiprogrammes	Multiprograms
متعدد المعالجة	Multitraitements	Multitreatment
الحاسب المتوازي	Ordinateur parallèle	Parallel computer

Parallelism	parallélisme	التوازي
Message passing	Passage de message	تمرير الرسائل
Hot spot	Point chaud	النقاط الساخنة
Procedure	Procédure	الإجراء
procedure	procedures	الإجرائيات
Processor	Processeur	المعالج
Cube network	Réseau cubique	الشبكة المكعبية
Dynamic network	Réseau dynamique	الشبكة الديناميكية
Toroidal network	Réseau en anneau	الشبكة الحلقية
Bus network	Réseau en bus	شبكة الناقل
Linear network	Réseau linéaire	الشبكة الخطية
Matrix network	Réseau matricielle	الشبكة المصفوفية
Static network	Réseau statique	الشبكة السكونية
Interconnection networks	Réseaux d'interconnection	شبكات الربط
Tree network	Réseaux en arbre	الشبكة الشجرية
Routers	Routeurs	الموجهات
supercomputer	Super- ordinateur	الحاسب الخارق
Concurrency	Synchronisation	التزامن
Expert system	Système expert	النظام الخبير
Mapping	cartographie	المقابلة

انجليزي	فرنسي	عربي
Edge	Arête	الضلع
Critical path	Chemin critique	المسار الحرج
Complexity	Complexité	التعقيد
Space complexity	Complexité espace	التعقيد في الحيز
Time complexity	Complexité temps	التعقيد الزمني
Decomposition	Décomposition	التقسيم
Data decomposition	Décomposition de données	تقسيم البيانات
Exploratory decomposition	Décomposition explorative	التقسيم الإستكشافي
Recursive decomposition	Décomposition récursive	التقسيم العودي
Specutative decomposition	Décomposition spéculative	التقسيم التخميني
Degree of synchronization	Degré de synchronisation	درجة التزامن
Granularity	Granularité	الحبوبية
Fine granularity	Granularité fine	الحبوبية الناعمة
Heavy granularity	Granularité lourde	الحبوبية الخشنة
Graph	Graphe	البيان
Dependency graph	Graphe de dépendance	مخطط التبعية
Undirected graph	Graphe non orienté	بيان غير موجه
Directed graph	Graphe orienté	بيان موجه
Interaction	Interaction	التفاعل

انجليزي	فرنسي	عربي
Tasks	Les taches	المهام
Adjancy matrix	Matrice d’adjacence	مصفوفة الجوار
node	Nœud	العقدة
Vertex	sommet	الرأس(القمة)
Bubble sort	Tri à bulles	الفرز الفقاعي
Odd–even transposition	Tri pair–impair	الإبدال الفردي–الزوجي
Quick sort	Tri rapide	الفرز السريع
Grid	grille	شبكة
Block	Bloc	كتلة
Thread	Processus léger(fil d'exécution)	تريد (خط تنفيذ)