République algérienne démocratique et populaire
الجمهورية الجزائرية الديمقراطية الشعبية

Ministère de l'enseignement supérieur et de la recherche scientifique
وزارة التعليم العالي والبحث العلمي

Université 8 mai 1945- Guelma

Faculté de Mathématiques, d'Informatique et de Sciences de la Matière

Département d'Informatique

Domaine : Mathématiques et Informatique

Filière : Informatique

Parcours : Système Informatique

Support de cours

Architectures parallèles

Réalisé par : Dr. BENHAMIDA Nadjette

Table des matières

Liste des figures

Liste des tables

Liste des algorithmes

Liste des acronymes

SIMD: Single Instruction stream Multiple Data stream.

SISD: Single Instruction stream Single Data stream.

MISD: Multiple Instruction stream Single Data stream.

MIMD: Multiple Instruction stream Multiple Data stream.

NUMA: Non-Uniform Memory Access.

CC-Numa: Cache-Coherent Non-Uniform Memory Access.

FI : Flux d'Instruction.

FD : Flux de Données.

CPU : Central Processing Unit.

UC : Unité de Contrôle.

UT : Unité de Traitement.

ML : Mémoire Locale.

CAO : conception assistée par ordinateurs.

GPU : Graphical Processing Unit.

MPI : Message Passing Interface.

PVM : Parallel Virtual Machine.

IBM : International Business Machines.

Avant-propos

Ce cours s'adresse aux étudiants de 2eme année Master de la filière « Informatique » spécialité « Système Informatique ». Il vise à introduire de nouvelles notions de base relatives aux architectures des ordinateurs, au parallélisme de traitements et à initier les étudiants à ces nouveaux concepts architecturaux. En effet, ce cours offre un certain nombre de modèles et outils permettant de résoudre un problème donné dans un environnement parallèle.

Ce polycopié représente une synthèse des cours assurés depuis 2014 à ce jour au sein du département d'informatique à l'université 08 Mai 1945 - Guelma.

Pré-requis : Le contenu de ce support de cours est présenté d'une manière simplifiée et n'exige pas des connaissances approfondies particulières. Les seules connaissances préalables qui peuvent être utiles sont celles reliées aux notions de bases des architectures des ordinateurs.

Introduction

L'architecture de von Neumann est un modèle d'ordinateur qui utilise une seule structure de stockage pour conserver à la fois les instructions et les données du calcul que ce soit entrées ou résultats. En traitant les instructions de la même façon que les données, un ordinateur qui a un programme stocké en mémoire peut facilement modifier les instructions. Cependant, cet objectif est devenu moins important avec l'apparition de l'utilisation de registres d'index et de l'adressage indirect en tant que caractéristique standard des processeurs actuels. Grace aux nouvelles architectures des ordinateurs, la modification à faible échelle des instructions du programme est devenue inutile car cela rendrait inefficaces les techniques de gestion de mémoire cache et de pipeline dans les processeurs modernes ; c'est exactement pourquoi l'architecture de type Von Neumann est devenue obsolète. Par conséquent, un nouveau type d'architectures d'ordinateurs a apparu ; dit « architecture parallèle ». En plus, les architectures parallèles sont devenues le paradigme dominant pour tous les ordinateurs depuis les années 2000.

L'objectif de ce polycopié de cours est de tenter de cerner le domaine des architectures des ordinateurs parallèles en présentant leurs concepts de base, leurs caractéristiques et leurs différentes classes selon la taxonomie de Flynn.

La première partie de ce polycopié présente l'historique de l'évolution des architectures des ordinateurs allant de la 1ère génération jusqu'à la 5ème. La deuxième partie est consacrée à l'organisation et les concepts de base des architectures parallèles.

Chapitre I : Historique et évolution des différentes architectures

1. Introduction

Il est important de comprendre que même si les premiers ordinateurs ont été achevés après la seconde guerre mondiale, leur conception repose sur le résultat de divers prototypes pendant plusieurs années. L'objectif de ce chapitre est de présenter l'historique et les principales étapes d'évolution des différentes architectures des ordinateurs.

Ce chapitre est organisé comme suit : tout d'abord un préhistorique des ordinateurs informatiques est détaillé dans la section 2. Dans la section 3, les différentes générations d'architecture sont définies en mettant l'accent sur leurs historiques, caractéristiques et une brève discussion de chacune entre eux. La section 4 est consacrée à la citation de quelques ordinateurs de différentes générations. Une série corrigée de questions de révision est donnée dans la section 5 alors qu'une conclusion est présentée dans la section 6.

2. Préhistorique

L'ère des ordinateurs modernes a commencé avec les grands développements de la seconde guerre mondiale. Plus précisément, les premières séries d'ordinateurs dites « séries-Z » qui ont été lancé par « Konrad Zuse » en 1938. Ils étaient des calculateurs électromécaniques comportant une mémoire et assuraient une programmation limitée. Le « Z1 » (ou *Zuse I*), n'a pas fonctionné correctement, mais il a donné à son inventeur l'expérience nécessaire pour achever d'autres modèles tels que le Z2 en 1939, le « Z3 » en 1941 et le « Z4 » en 1945 [1] [2] [3].

Durant la même période, « John Vincent Atanasoff » et « Clifford E. Berry », de l'université de l'État de l'Iowa, ont conçu l'ordinateur « Atanasoff-Berry » en 1937, un additionneur à 16 bits qui a été testé avec succès en 1942. En 1940, « George Stibitz » et « Samuel Williams » ont développé le « Complex Number Calculator » (ou Model I) aux « Bell Laboratories », un calculateur à base de relais téléphoniques. Ce dernier est considéré comme la première machine contrôlée à distance via une ligne téléphonique et qui permet de réaliser une multiplication en seulement une seule

minute. En 1944, le « Harvard Mark I » a été inventé par « Howard Aiken » au niveau de « IBM » [1] [2] [3]. Ce dernier est appelé aussi selon « International Business Machines » (IBM) « Automatic Sequence Controlled Calculator » (ASCC, Figure1).

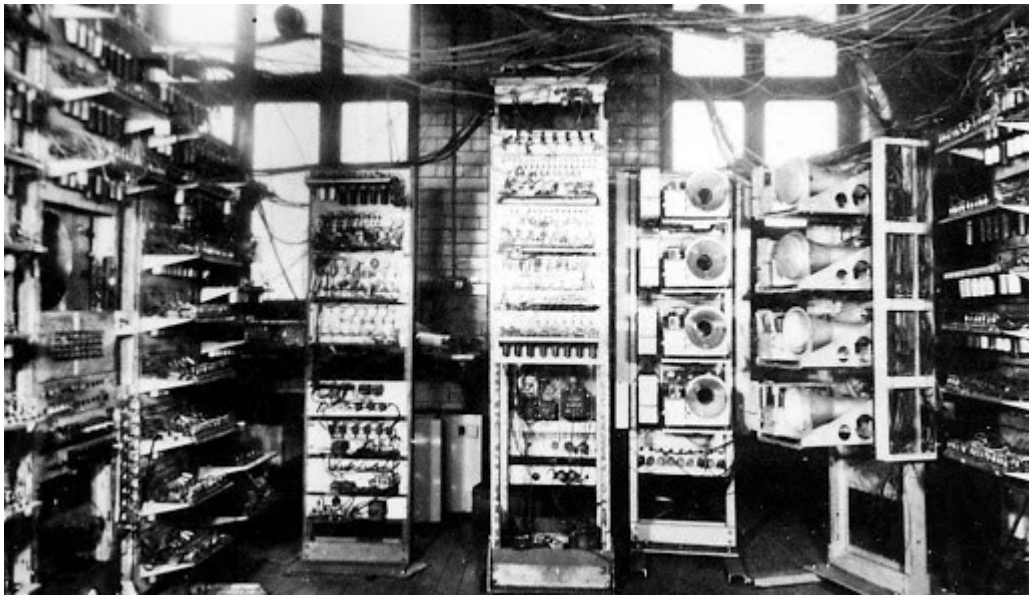


Figure 1 : Le calculateur Mark I [w1].

3. Ordinateur et changements technologiques

Le développement des architectures des ordinateur a passé par plusieurs générations à savoir : (1) la première génération, (2) la deuxième génération, (3) la troisième génération, (4) la quatrième génération et (5) la cinquième génération.

3.1 Première génération (1945-1957)

3.1.1 Historique

Avec la fin de la seconde guerre mondiale, plusieurs ordinateurs ont vu le jour. Parmi les premiers ordinateurs de cette génération, on peut citer l'« Electronic Numerical Integrator and Computer» (ENIAC, Figure 2 [4]) achevé en 1944 par « Presper Eckert » et « John William Mauchly » et son successeur l' « Electronic Discrete Variable Automatic Computer» (EDVAC, Figure 3) achevé en 1945 et l' « UNIVersal Automatic Computer I» (UNIVAC I) en 1951 [1].

En 1946, alors que le projet EDVAC était en cours, un projet similaire a été lancé à l'université de Cambridge. Il s'agissait de construire un ordinateur à « programme enregistré », connu sous le nom de « Electronic Delay Storage Automatic Calculator »

(EDSAC). C'est en 1949 que l'EDSAC est devenu le premier ordinateur complet, à programme enregistré et entièrement opérationnel [1].

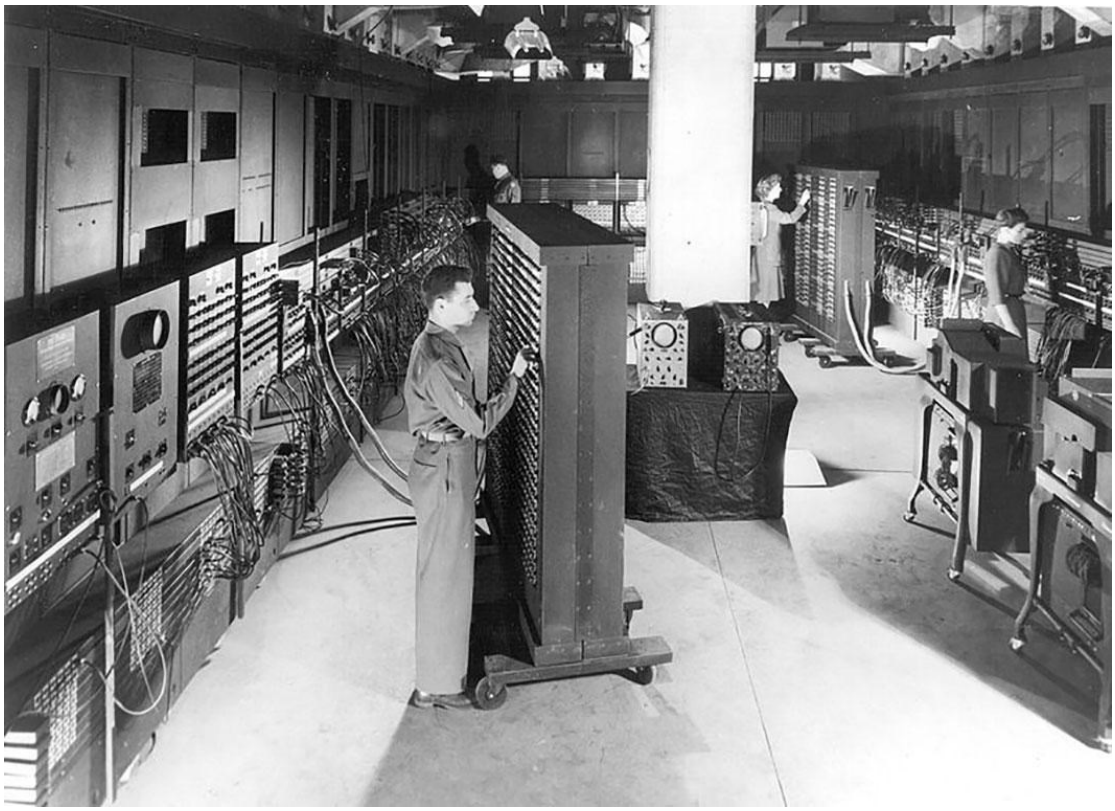


Figure 2 : Une vue partielle de l'ENIAC [4].

L'achèvement de l'EDSAC a donné lieu à une série de machines présentées à Harvard composée de MARK I, II, III et IV. Les deux dernières machines ont introduit le concept de mémoires séparées pour les instructions et les données. Il faut noter que, aujourd'hui, le terme "architecture de Harvard" est utilisé pour décrire les machines dotées de caches séparés pour le stockage des instructions et des données [1].

Figure 3 : Une vue partielle de l'EDVAC [w2].

Cependant, les premières machines utilisant l'architecture de von Neumann ont été apparu à partir de 1948. Par conséquent, et contrairement à toutes les ordinateurs précédentes, les programmes étaient stockés dans la même mémoire que les données et pouvaient, ainsi, être manipulés comme des données. La première machine qui utilisait cette architecture est le « Small-Scale Experimental Machine » (SSEM) construit à l'université de Manchester en 1948 [2] [3]. La table 1 illustre quelques généralités sur ordinateurs de la première génération.

3.1.2 Caractéristiques

La première génération était caractérisée par [4] :

- Ordinateurs à cartes perforées (Figure 4.) et à bandes magnétiques.
- Utilisation de tubes à vide (Figure 5).
- Programmation physique en langage machine.
- Chaque ordinateur avait son propre jeu de codes binaires, appelé langage machine, qui lui indiquait comment fonctionner.
- Les instructions correspondaient spécialement à la tâche pour laquelle ces ordinateurs avaient été construit.
- Ces ordinateurs ont été destiné notamment pour le calcul numérique.

- Appareils immenses, lourds et consommaient une énergie élevée.
- Prix élevé par rapport à leurs capacité et performance.

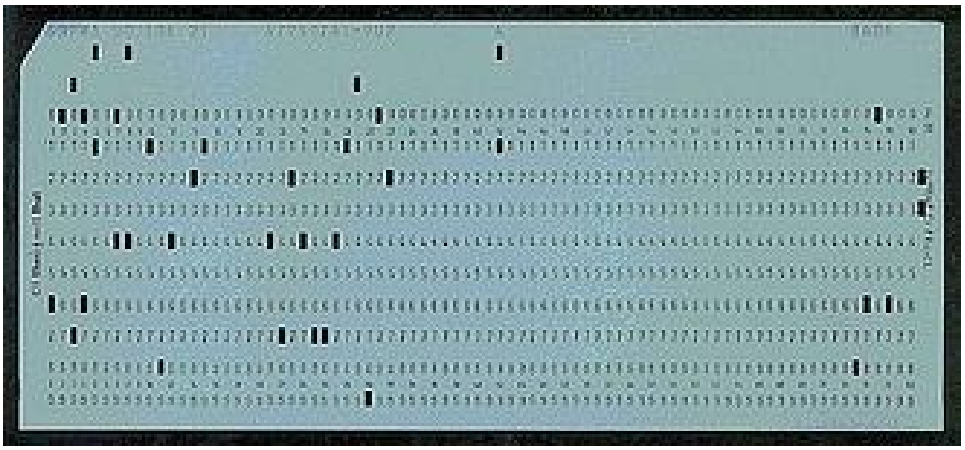


Figure 4 : Carte perforée [w3].

Figure 5 : Tubes à vide [w4].

3.1.2 Discussion

Toutes ces caractéristiques ont rendu les ordinateurs de cette génération difficiles à programmer et ont limité leur souplesse et leur vitesse.

3.2 Deuxième génération (1957- 1963)

3.2.1 Historique

A partir de 1956, les transistors ont remplacé avantageusement les tubes à vide. Depuis, la taille des ordinateurs n'a cessé de diminuer. En plus, grâce aux dernières avancées technologiques concernant la mémoire à tores magnétiques, les ordinateurs de la seconde génération étaient plus petits, plus rapides, plus fiables et consommaient moins d'énergie par rapport aux ordinateurs de la première génération. Mais comme les premiers ordinateurs de cette génération ont été spécialement créés pour effectuer des taches bien précises des laboratoires et des secteurs financiers, elles étaient très chères, et trop puissantes pour les faibles exigences de ces secteurs par rapport aux machines de la première génération [5].

Jusqu'aux années 60, il y avait un certain nombre d'ordinateurs de la seconde génération ayant connu un certain succès commercial, notamment dans le secteur financier, les universités ou le gouvernement. Ces ordinateurs ont utilisé des périphériques courants pour les ordinateurs modernes tels que : une imprimante, un enregistreur à bande magnétique, des disques magnétiques, une mémoire, un système d'exploitation et les programmes ont été stocké dans la mémoire de l'ordinateur. Parmi les plus importants ordinateurs de cette génération qu'on puisse citer, on trouve le « Stretch » d'IBM et le « LARC » de Sperry-Rand et notamment l' « IBM 1401 », qui a été mondialement accepté par l'industrie, et qui est souvent considéré comme étant le modèle de l'industrie informatique [5].

3.2.2 Caractéristiques

Les ordinateurs de la seconde génération ont été caractérisées par [4] :

- Utilisation de transistors (Figure 6) ce qui a augmenté la fiabilité.
- Utilisation de mémoires de masse pour le stockage périphériques.
- L'assembleur a remplacé le langage machine.
- Temps d'accès moyen de l'ordre de la microseconde.
- Fonctionnement séquentiel des systèmes de programmation (langages évolués) tel que FORTRAN Mainframes.

Figure 6 : Les transistors utilisés pour les machines de la seconde génération [w5].

3.2.3 Discussion

L'enregistrement des programmes dans la mémoire centrale signifie qu'une instruction pouvait être remplacée directement par une autre en vue d'effectuer une autre fonction. Aussi, l'utilisation des langages de haut niveau comme les Cobol (COmmon BusinessOriented Language) et FORTRAN (FORmula TRANslator) a permis de remplacer le langage machine et l'assembleur par des mots, des phrases et des formules mathématiques beaucoup plus proches du langage naturel, ce qui a facilité la programmation de ces ordinateurs.

Par conséquent, les ordinateurs ont eu la flexibilité nécessaire pour devenir utilisables particulièrement dans les secteurs financiers et de nouveaux métiers sont émergés tels que : le programmeur, l'analyste et l'expert aussi l'industrie du logiciel a été bien développé avec les ordinateurs de la seconde génération.

3.3 Troisième génération (1963 - 1971)

3.3.1 Historique

Malgré que l'intégration des transistors représente une amélioration significative par rapport à l'utilisation des tubes à vide, ils ont un autre problème qui est la génération de beaucoup de chaleur en consommant plus d'énergie et cela a endommagé les parties internes sensibles de l'ordinateur. C'est pourquoi un ingénieur de « Texas Instruments » nommé « Jack Kilby » a développé ce qu'on appelle aujourd'hui « le circuit intégré »

en 1968 (Figure7) à base de quartz pour résoudre ce problème [5]. Le premier circuit intégré combinait trois composants électroniques sur un petit disque de silicium.

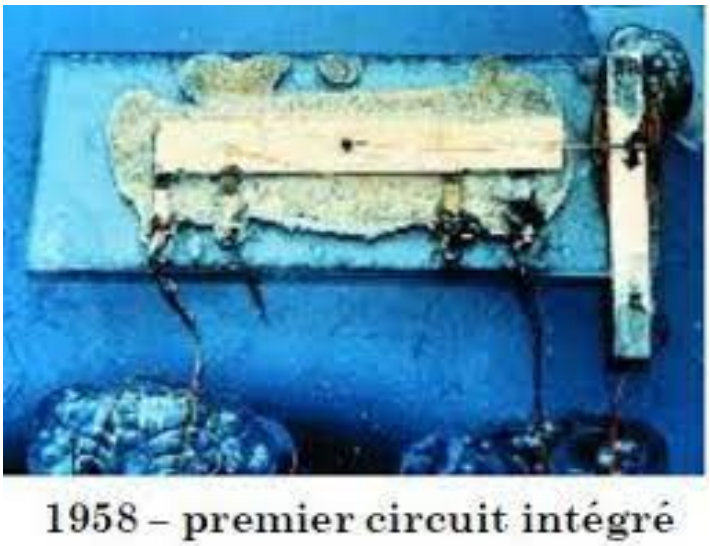


Figure 7: Les premiers circuits intégrés [w5].

Le premier ordinateur utilisant des circuits intégrés est IBM 360 fabriqué en 1965 (Figure 8).

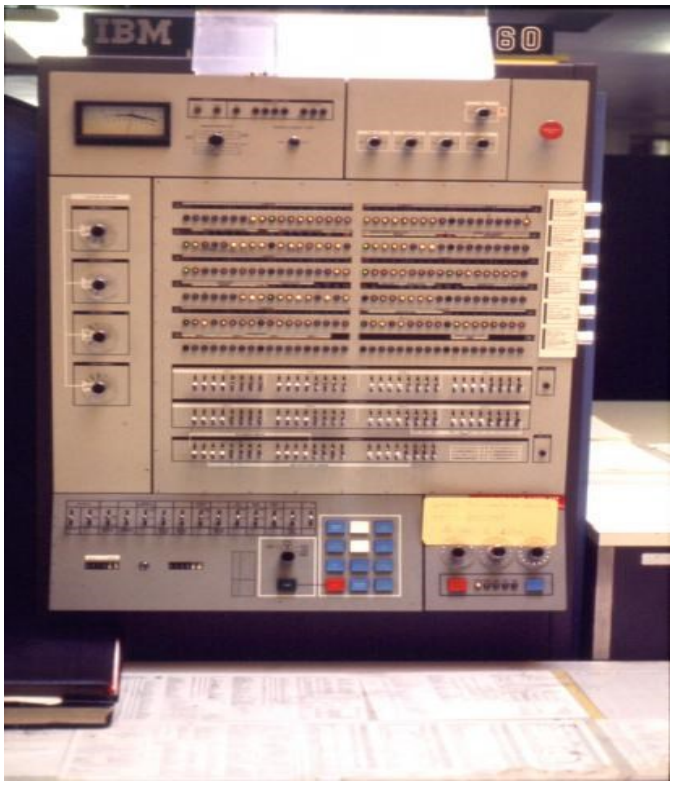


Figure 8 : L'IBM 360 de INRA en 1980 [w6].

3.3.2 Caractéristiques

Généralement les ordinateurs de la troisième génération sont caractérisés par [4] :

- L'intégration des circuits intégrés.
- Les ordinateurs ont devenu de plus en plus petits, puisque plus de composants étaient mis sur un seul circuit intégré.
- L'utilisation d'un système d'exploitation (OS - operating system) a permis de faire tourner plusieurs programmes différents sur une même machine avec un programme central contrôlant et coordonnant la mémoire centrale de l'ordinateur.

3.3.3 Discussion

La troisième génération est particulièrement marquée par la distribution de l'ordinateur, créant un nouveau marché, celui du grand public parce que l'intégration des circuits intégrés a permis de réduire considérablement le coût du matériel. Cependant, les ordinateurs étaient toujours énormes, surpuissants et centralisent les traitements.

Aussi, les recherches s'orientent vers l'élaboration de langages de programmation de plus en plus proches des langues naturelles, avec notamment l'apparition de langages puissantes tels que : PL/1, Pascal, etc.

3.4 Quatrième génération (1971- aujourd'hui)

3.4.1 Historique

Après les circuits intégrés, les principales améliorations à réaliser tournent autour de la réduction de leurs tailles. Par conséquent, plusieurs techniques ont été apparues comme Le LSI (Large Scale Integration), le VLSI (Very Large Scale Integration) et l'ULSI (Ultra Large Scale Integration). Ces derniers permettent de placer plusieurs centaines, centaines de milliers ou millions, respectivement, de composantes sur le même support [5].

Le premier microprocesseur est le « Intel 4004 » qui a été développé en 1971 (Figure 9). Il était le premier circuit intégré incorporant tous les éléments d'un ordinateur dans un seul boîtier : unité de calcul, mémoire, contrôle des entrées/sorties. Alors qu'il fallait avant plusieurs circuits intégrés différents, chacun dédié à une tâche particulière, un seul microprocesseur pouvait assurer autant de travaux différents que possible.

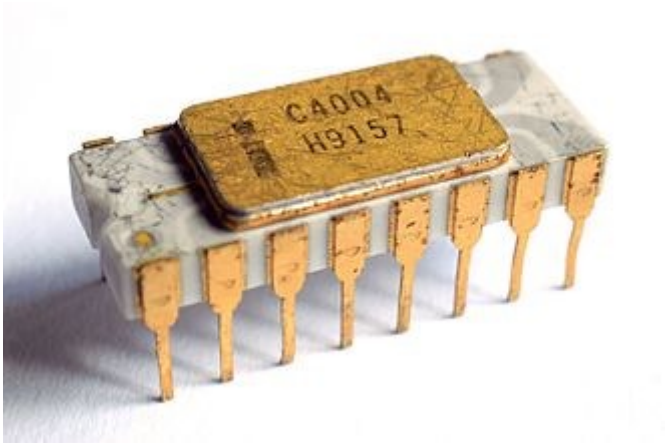


Figure 9 : Le premier microprocesseur commercialisé « Intel 4004 » [w7].

En 1981, IBM a lancé son premier ordinateur personnel (Personnal computer, Figure 10), à utiliser à la maison, au bureau, dans les écoles, etc.

Figure 10 : L'ordinateur personnel « IBM PC » , modèle 5150 [w7].

3.4.2 Caractéristiques

Parmi les caractéristiques des ordinateurs de la quatrième génération, on peut citer [4] :

- Réduction extrême des composants.
- Apparition des microprocesseurs.
- Discrimination des champs d'application.
- Apparition de la micro-informatique.
- L'aspect logiciel prend le pas sur l'aspect matériel.

De même, on peut rajouter :

- Des circuits de plus en plus de petite taille.
- Meilleures performances, puissance et fiabilité.

- L’apparition des réseaux informatiques.
- La création des langages simplifiés et évolués de programmation.

3.4.3 Discussion

La taille réduite des circuits intégrés a permis de réduire la taille et le prix des ordinateurs ainsi qu’améliorer leurs performances, leur puissance, et également leur fiabilité. C'est l’apparition de la micro-informatique et les ordinateurs ont continué de diminuer en termes de taille, jusqu'à la création des ordinateurs portables tenant dans une sacoche, puis des ordinateurs tenant dans la main. Cependant, la construction des gros ordinateurs n'est pas abandonnée.

En plus, les ordinateurs de la quatrième génération pouvaient être reliés entre eux ou mis en réseau en utilisant soit un réseau local dit aussi LAN (Local Area Network) ou un réseau étendu, appelé WAN (Wide Area Network) pour partager de la mémoire, des logiciels, des périphériques, des informations ou même pour communiquer entre eux. De cette façon, ils pouvaient également distribuer le temps machine entre plusieurs terminaux dans l’objectif d’améliorer leurs performances.

3.5 Cinquième génération (1982-aujourd’hui)

3.5.1 Historique

Les exigences de performance des ordinateurs de cinquième génération ont été mises en lumière lors de la sixième Conférence Internationale sur le Génie Logiciel (Tokyo 1982. En effet, ce terme était une initiative du ministère japonais du Commerce international et de l’Industrie (MITI), lancée en 1982, pour créer des ordinateurs utilisant le calcul massivement parallèle et la programmation logique. Le logiciel de base pour les ordinateurs de cinquième génération est le reflet direct de la structure des systèmes d'application prévus et comprend l'interface intelligente, les systèmes de résolution de problèmes et d'inférence et les systèmes de gestion de bases de connaissances. Les caractéristiques de ces ordinateurs restent encore imprécises, mais elles s'orientent vers des machines « intelligentes » qui devraient faire beaucoup de traitements en parallèle, utiliser de façon importante des systèmes intelligents basés sur des connaissances, et posséder des interfaces en langage naturel [6].

A l'issue du projet de MITI, divers prototypes de machines séquentielles et parallèles ont été construits et des logiciels de base ont été développés et testés avec succès. En

ce qui concerne le matériel, la première machine séquentielle à inférences logique, est « Personal Sequential Inference I » (PSI-I) et ses versions avancées : « PSI-II » a été réalisée en 1986 et PSI-III a été achevée en 1990. Ces modèles ont été construits concurremment par Mitsubishi et Oki. En ce qui concerne les machines parallèles, plusieurs modèles dits « Parallel Inference Machine » (PIM) ont été construits dont le plus puissant est PIM/p, construit par Fujitsu, début 1992 [7] [8].

3.5.2 Caractéristique

Définir les caractéristiques des ordinateurs de la cinquième génération est plutôt difficile, puisqu'ils sont toujours en train de se développer. Toutefois, on peut citer quelques particularités de ces ordinateurs [4] :

- Réduction de la taille des composants poussée à l'extrême.
- Apparition de nouvelle architecture physique dite architecture parallèle.
- Machine à processeurs parallèles.
- Possibilité de choix d'ordre des vecteurs séquentiels à traiter.
- Vitesse de traitement augmentée.
- Nouvelles structures et représentations des données.
- Apparition de l'intelligence artificielle.
- Ajout du traitement de l'aspect sémantique à celui de l'aspect syntaxique de l'information.
- Ordinateurs à photons (ordinateurs photoniques ou encore optiques).

3.5.3 Discussion

Plusieurs améliorations dans la construction des ordinateurs et la technologie en général ont été faits pendant cette génération. Une de ces améliorations concerne le calcul parallèle, qui remplace la simple unité de traitement décrite par von Neumann par un ensemble de processeurs travaillant en parallèle pour résoudre le même problème. Une autre amélioration est la technologie des supraconducteurs, qui éliminent la résistance à la conductivité électrique et permettent d'améliorer les vitesses de transmission de l'information. Également, en utilisant les dernières avancées technologiques, les ordinateurs de la cinquième génération pourraient comprendre le langage naturel et simuler la pensée humaine. Par exemple, on se basant sur les techniques de l'intelligence artificielle, des systèmes experts aident les médecins à réaliser des

diagnostics en appliquant les prescriptions qu'un médecin utiliserait pour répondre aux besoins d'un patient [5].

4. Exemples d'ordinateurs de la première génération

Ordinateur	Année	Pays	Développeur	Description
Zuse 3	1941	Allemagne	Konrad Zuse	- La première machine programmable absolument automatique. - Utilisé par l'institut de recherche aéronautique allemand. - Il est destiné à concevoir des avions et des missiles.
Zuse 4	1945	Allemagne	Konrad Zuse	- Un ordinateur électromécanique multi-usage programmable. - Il a été livré à l'institut de mathématiques appliquées de l'École polytechnique fédérale de Zurich, en Suisse en 1950 pour faire les calculs du barrage de la Grande-Dixence. - Ensuite il a été transféré à l'Institut franco-allemand de recherches de Saint-Louis en France.
Harvard Mark I	1944	États-Unis	Howard Aiken – IBM	- Un ordinateur à relais électromécanique et à programme externe. - Utilisée par la marine américaine. - Essentiellement destiné pour le calcul scientifique.

Harvard Mark II	1947	Royaume-Unis	Howard Aiken (université de Harvard)	- Il a été financé par la marine des États-Unis. - Il est destiné au décodage des messages secrets allemands par les Anglais. - Après une série de tests il a été livré au centre de test de l'US Navy en 1947.
Harvard Mark III	1949	Royaume-Unis	Howard Aiken (université de Harvard)	- Il était partiellement électronique et partiellement électromécanique. - Il a été construit pour la United States Navy.
Harvard Mark IV	1952	Royaume-Unis	Howard Aiken (université de Harvard)	- Il était l'un des premiers ordinateurs complètement électroniques. - Il s'agissait d'un ordinateur à programme enregistré. - Il a été toujours l'armée de l'air américaine.
ENIAC	1945	États-Unis	John Presper Eckert & John W. Mauchly	- Le premier ordinateur entièrement électronique. - Il a été financé par le gouvernement américain et développé à l'Université de Pennsylvanie. - L'ENIAC opère en mode décimal.
EDVAC	1945	États-Unis	J. Presper Eckert, John William Mauchly, John von Neumann et all.	- Il utilise une mémoire qui servait à la fois au stockage du programme et des données. - L'élément-clef de son architecture était l'unité centrale de calcul. - Un ordinateur électronique. - Il opère en mode binaire.

UNIVAC I	1951	États-Unis	J. Presper Eckert et John Mauchly,	- L'un des premiers ordinateurs commercialisés à prendre avantage de toutes les découvertes. - Le premier ordinateur est livré à l'United States Census Bureau le 30 mars 1951.

Table 1 : Exemples d'ordinateurs.

5. Questions de révision

Question 1 : Quelles sont les 5 générations de l'ordinateur ?

Question 2 : Quand sont apparus les premiers ordinateurs ?

Question 3 : Quel est le premier ordinateur au monde ?

Question 4 : Comparer les différentes générations des architectures des ordinateurs en spécifiant l'intervalle du temps, au moins un exemple de machine et les principales caractéristiques de chaque génération.

Solutions :

Question 1 :

Les 5 générations de l'ordinateur sont :

- Première génération. 1945-1957.
- Deuxième génération. 1957-1963.
- Troisième génération. 1963-1971.
- Quatrième génération. 1971- 1982.
- Cinquième génération : 1982-aujourd'hui.

Question 2 :

Les premiers ordinateurs sont apparus après la seconde guerre mondiale bien précisément, le premier ordinateur entièrement électronique est le ENIAC achevé en 1945. Cependant, il est important de comprendre que leur conception était lancée bien avant et elle repose sur le résultat de divers prototypes pendant plusieurs années.

Question 3 :

Le premier ordinateur entièrement électronique est le ENIAC achevé en 1945. Il a été conçu par John Presper Eckert et John W. Mauchly à l'Université de Pennsylvanie et financé par le gouvernement américain

Question 4 :

Génération	Période	Machine	Caractéristiques
Première génération	1945 à 1957	Mark I	• Ordinateur à cartes perforées et à bandes magnétiques • Programmation physique en langage machine • Calcul numérique. • Appareils immenses, lourds, énergie élevée. • Utilisation de tubes à vide. • Prix élevé / capacité et performance.
Deuxième génération	De 1957 à 1963	IBM 1401	• Utilisation de transistors et donc l'augmentation de la fiabilité • Utilisation de mémoires de masse pour le stockage périphériques. • Temps d'accès moyen (de l'ordre de la microseconde). • Fonctionnement séquentiel des systèmes de programmation (langages évolués)
Troisième	1964 à 1971	IBM 360	• Utilisation des circuits intégrés permettent de réduire considérablement le coût du matériel. • La distribution de l'ordinateur, créant un nouveau marché, celui du grand public. • Orientation vers l'élaboration de langages de plus en plus proches des langues naturelles, avec notamment l'apparition de langages puissantes tels que : PL/1, Pascal (1970), etc. • Cependant, les ordinateurs étaient énormes, surpuissants et centralisent les traitements.

Quatrième	1971 -1982	DIEHL Alphatronic	• Réduction extrême des composants. • Apparition des microprocesseurs. • Discrimination des champs d'application. • Apparition de la micro-informatique. • L'aspect logiciel prend le pas sur l'aspect matériel. • Des circuits de plus en plus de petite taille. • Meilleures performances, puissance et fiabilité. • L'apparition des réseaux informatiques. • La création des langages simplifiés et évolués de programmation.
Cinquième génération	1982-nos jours	Parallel Inference Machine/p	• Réduction de la taille des composants poussée à l'extrême. • Apparition de nouvelle architecture physique dite architecture parallèle. • Machine à processeurs parallèles. • Possibilité de choix d'ordre des vecteurs séquentiels à traiter. • Vitesse de traitement augmentée jusqu'au gigalips. • Nouvelles structures et représentations des données. • Apparition de l'intelligence artificielle. • Ajout du traitement de l'aspect sémantique à celui de l'aspect syntaxique de l'information. • Ordinateurs à photons (ordinateurs photoniques ou encore optiques).

Table 2 : Comparaison des différentes générations des ordinateurs.

6. Conclusion

Dans ce chapitre, les principaux acteurs qui ont attribué à l'évolution des architectures des ordinateurs ainsi que leurs différentes générations et quelques exemples d'architectures ont été présenté.

Les architectures des processeurs parallèles seront présentées dans le chapitre suivant.

Chapitre II Organisation et concepts des architectures parallèles

1. Introduction

Le domaine du traitement parallèle concerne les méthodes architecturales et algorithmiques permettant d'améliorer les performances des ordinateurs numériques ou d'autres critères tels que la rentabilité et la fiabilité grâce à diverses formes de concurrence. Même si le calcul concurrent existe depuis les premiers jours des ordinateurs numériques, ce n'est que récemment qu'il a été appliqué d'une manière permettant d'obtenir de meilleures performances, ou un meilleur rapport coût-efficacité, par rapport aux superordinateurs habituels. Comme tout autre domaine scientifique ou technologique, l'étude des architectures et algorithmes parallèles nécessite une motivation, une vue d'ensemble montrant les relations entre les problèmes et les différentes approches pour les résoudre, ainsi que des modèles pour comparer, connecter et évaluer les nouvelles idées, dont l'objectif de ce chapitre.

Ce chapitre est organisé comme suit : tout d'abord la définition des architectures parallèles, leurs principales caractéristiques et leurs classifications sont discutés dans la section 1. L'architecture SIMD est présentée dans la section 2 en spécifiant ses aspects architecturaux, son approche de programmation et en donnant à la fin un exemple illustratif. De même, l'architecture MISD est détaillée dans la section 3 et MIMD dans la section 4. Une série de questions de révision est donnée dans la section 5 alors qu'une conclusion est présentée dans la section 6.

1.1 Définitions

Dans les architectures des ordinateurs, on peut définir une machine parallèle comme étant, essentiellement, une machine qui possède un ensemble de processeurs qui coopèrent et communiquent ensemble [4] et qui concourent dans le but d'exécuter un programme parallèle. Par conséquent, on peut définir les ordinateurs parallèles comme étant des machines qui comportent une architecture parallèle.

Un programme parallèle est un ensemble fini de processus séquentiels communiquant entre eux par échange de messages.

La performance d'une architecture parallèle est la combinaison des performances de ses ressources tels que la latence, le débit, leur coût de mise en œuvre, etc.

1.2 Pourquoi le parallèlisme ?

La réponse à cette question se résume dans les trois points suivants [9] :

- Les besoins croissants en hautes performances (voir [10], [11]),
- L'évolution de la technologie, et
- L'évolution des applications.

Deux facteurs majeurs contribuent à l'amélioration des performances des processeurs modernes [9] [12] : (1) La technologie rapide des circuits et les caractéristiques architecturales, à savoir des caches de plus en plus grands, des bus multiples de plus en plus rapides, le pipelining, les architectures superscalaires, etc.

Le deuxième facteur concerne le fait que (2) Les calculateurs avec une seule unité centrale de traitement (*central processing unit* CPU) sont souvent incapables de répondre à certains besoins comme : l'analyse des flots de liquides et aérodynamique, la simulation de systèmes complexes (physique, économie, biologie, météo, technique, etc.), l'analyse de données massives pour la prise de décisions stratégiques, la conception assistée par ordinateurs (CAO) et le Multimédia, etc. Du fait que ces applications sont caractérisées par une très grande quantité de calculs numériques et/ou une grande quantité de données en entrée (Figure 11).

L'une des solutions à ce besoin en performances est la création des architectures dans lesquelles plusieurs CPUs fonctionnent dans le but de résoudre une application donnée [9] [10]. De tels calculateurs ont été organisés de différentes manières selon certaines caractéristiques clés, notamment le nombre et complexité des CPU individuels, la disponibilité de mémoires partagées ou communes, la topologie d'interconnexion, les performances des réseaux d'interconnexion, les dispositifs d'Entrées/Sorties, etc.

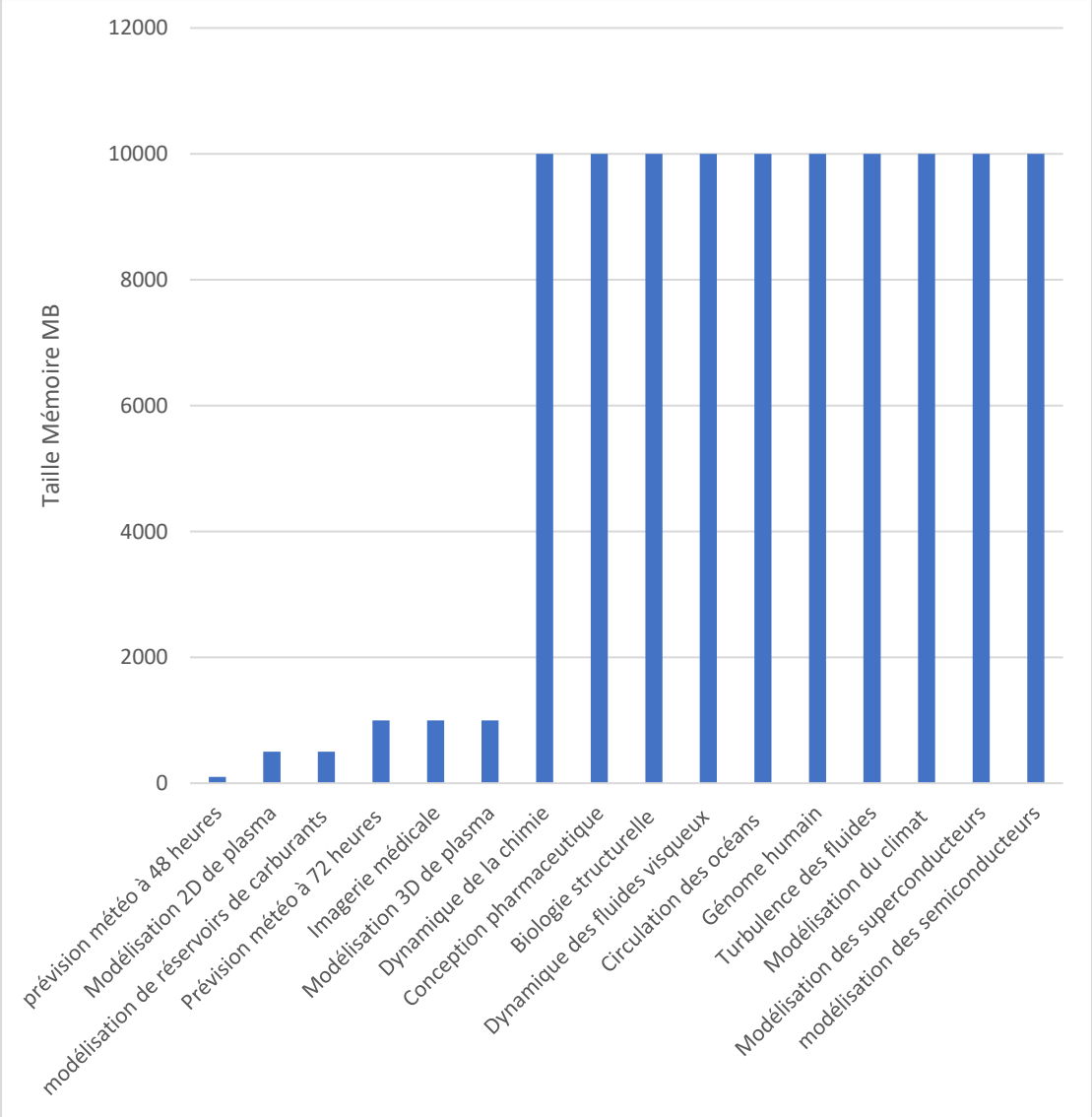


Figure 11 : Les besoins en performances [9].

Les architectures parallèles sont caractérisées par une ou plusieurs caractéristiques parmi les suivantes :

- Espace mémoire étendu.
- Plusieurs processeurs.
- Traitement partiel indépendant.
- Accélération des calculs complexes ou coûteux en temps d'occupation CPU.
- Calcul parallèle répétitif sur un large ensemble de données structuré.

Alors que les performances architectures parallèles sont limités lorsqu'il existe dans le programme parallèle :

1. **Des dépendances fonctionnelles :** que ce soit des dépendances de données, ou des dépendances de contrôle.
2. **Des dépendances de ressources :** si le nombre de processeurs est insuffisant par rapports le nombre des taches parallèles indépendantes.
3. **Des délais de communication élevés :** les communications peuvent dans ce cas augmenter le temps d'exécution et donc il ne sera pas avantageux de paralléliser une application dans ce cas.

1.3 Taxonomie de Flynn

La taxonomie de Flynn est une classification des architectures d'ordinateur, proposée par Michael Flynn en 1966. Cette classification est basée sur [12] [13]: *(1)* la nature du flux d'instructions exécuté par le calculateur et (2) la nature du flux de données sur lesquels opèrent les instructions (Table 3).

Classification de Flynn		Flux des données (D)	
		Unique (S : single)	Multiple(M)
Flux des instructions (S)	Unique (S : single)	SISD	SIMD
	Multiple (M)	MISD	MIMD

Table 3 : Classification de Flynn [12] [13].

Par conséquent, nous obtenons quatre classes de processeurs dites la classe SISD, la classe SIMD, la classe MISD et la classe MIMD. Chaque classe peut avoir plusieurs autres types de processeurs qui peuvent être classés selon un ou plusieurs critères (Figure 12).

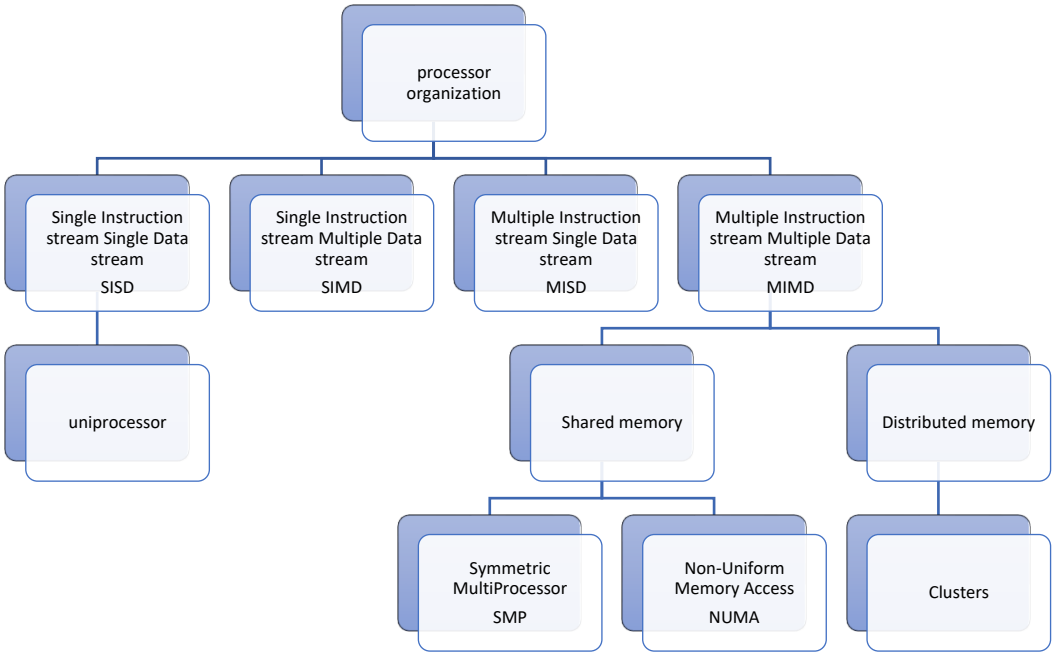


Figure 12 : Les organisations de processeurs [9].

SISD est identique à l'architecture de Von-Neumann avec un processeur unique qui exécute un seul flux d'instructions pour opérer sur une donnée stockée dans une mémoire unique [9]. Notons que les machines monoprocesseurs répondent au schéma suivant (Figure 13).

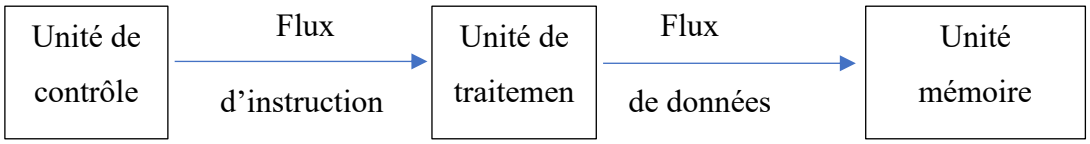


Figure 13 : L'architecture SISD.

1.4 Paradigme de la programmation parallèle

La programmation parallèle consiste, en général, à décomposer un seul problème de taille N en deux ou plusieurs sous-problèmes de petites tailles [12] qui ne présentent pas le même degré de parallélisme [14]. Aussi, elle consiste à utiliser la concurrence, la coordination et la communication dans un programme, afin d'augmenter la taille des problèmes qui peuvent être résolus ou de réduire le temps de réponse pour résoudre un problème donné.

Le modèle de programmation parallèle dépend fortement de l'architecture parallèle utilisée car nous ne pouvons pas programmer, par exemple, une machine avec une mémoire distribuée de la même manière qu'une machine avec une mémoire partagée.

Aussi, nous ne pouvons pas appliquer les principes de la programmation séquentielle en programmation parallèle. Parce qu'en programmation parallèle, le programmeur doit décider s'il applique le parallélisme de données (dans le cas d'une architecture SIMD par exemple), ou s'il opte pour un parallélisme d'instructions (MISD) ou encore s'il applique les deux parallélismes simultanément (MIMD).

Cependant, ce type de programmation pose quelques problèmes liés à la communication et la synchronisation de tâches parallèles suite à la charge d'un réseau par exemple ou encore à la différence des distances entre les lieux d'exécution des tâches parallèles.

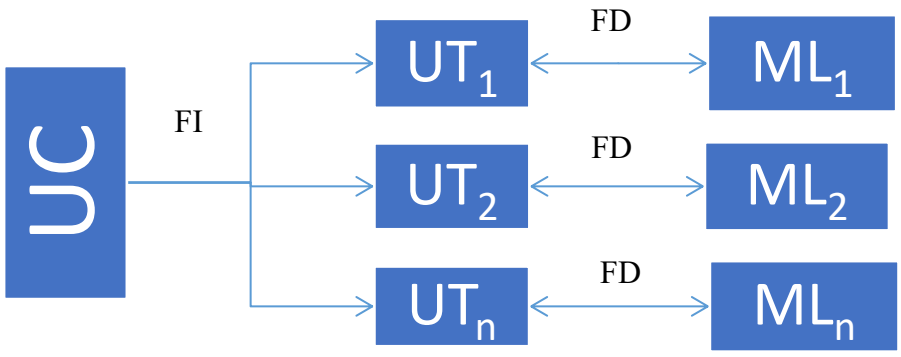
Les deux modèles de la programmation parallèles les plus connus sont :

- Paradigme de programmation Multithreading (section 2.2.1).
- Paradigme message-passing (section 3.2.1).

2. L'architecture SIMD

L'architecture SIMD est une architecture dont toutes les unités de traitement exécutent le même flux d'instructions qui opèrent sur des flux de données différents (Figure 14) [1] [9].

2.1 Aspects architecturaux



UC: Unité de Contrôle. UT: Unité de traitement. ML: Mémoire Locale.

FI: Flux d'instruction. FD: Flux de Données.

Figure 14 : L'architecture SIMD.

Cette architecture possède une seule unité de contrôle et plusieurs unités de traitement qui sont, en général, très simples avec une UAL qui exécute l'instruction transmise par l'unité de contrôle, quelques registres, et une mémoire locale. Les unités de calcul totalement synchronisées. Chaque unité d'exécution exécute la même instruction et chaque unité de traitement calcule un élément du résultat. L'architecture SIMD est caractérisée par un seul cycle d'horloge par traitement et un seul compteur ordinal. Une instruction unique contrôle l'exécution simultanée de plusieurs éléments de calcul. Notons que, les unités d'exécution parallèles sont synchronisées et chaque unité a son propre registre d'adresses [9]. Les processeurs vectoriels sont souvent considérés comme des architectures de type SIMD.

Le premier calculateur SIMD est ILLIAC IV crée aux années 70, avec 64 processeurs relativement puissants. Un autre exemple des machines actuelles est le Connection-Machine (CM2) avec 65536 unités de traitement très simples connectées en hypercube. La Table 4 illustre quelques exemples de processeurs SIMD [9].

Institution	Nom	Nb Max de processeur	FPUs	Bits par processeur	Bande passante (MB/s)	Année
Univ. Illinois	Illiac IV	64	64	64	2560	1972
Thinking machines	CM-2	65536	2048	1	16384	1987
Maspar	MP-1216	16384	0	4	23000	1989

Table 4 : Quelques exemples de machine SIMD [9].

Les principaux avantages de SIMD sont :

- Meilleur avec le parallélisme à grand volume de données;
- Amortit le coût du contrôle à travers les nombreuses unités d'exécution;
- Réduit la taille de la mémoire de programme;

L'inconvénient principal de l'architecture SIMD est :

- Peu adaptés à des instructions très variées (ex: switch).

Les processeurs vectoriels [9] : Ils sont des calculateurs SIMD qui peuvent opérer sur des vecteurs en exécutant simultanément la même instruction sur des paires d'éléments vectoriels et, donc, chaque instruction est exécutée par un élément de calcul séparé.

Cependant, des différentes architectures d'ordinateurs ont implémenté des opérations vectorielles en utilisant le parallélisme offert par les unités fonctionnelles pipelinées, de telles architectures sont nommées « processeurs vectoriels ». Ces derniers ne sont pas des processeurs parallèles parce qu'ils ne sont pas des CPU multiples s'exécutant en parallèle mais plutôt des processeurs SISD sur lesquels des instructions vectorielles sont implémentées sur des unités fonctionnelles en pipeline.

Les processeurs vectoriels ont généralement des registres vectoriels qui peuvent stocker de 64 à 128 mots chacun. Ils exécutent des opérations sur des scalaires et des instructions opérant sur des vecteurs (Figure 15). L'exécution d'une instruction vectorielle se résume dans les étapes suivantes :

- Chargement de vecteur depuis la mémoire vers le registre vectoriel.
- Rangement d'un vecteur en mémoire.
- Opérations arithmétiques et logiques entre des vecteurs.
- Opérations entre des vecteurs et des scalaires.

La figure 15 présente l'architecture simplifié d'un processeur vectoriel [15] :

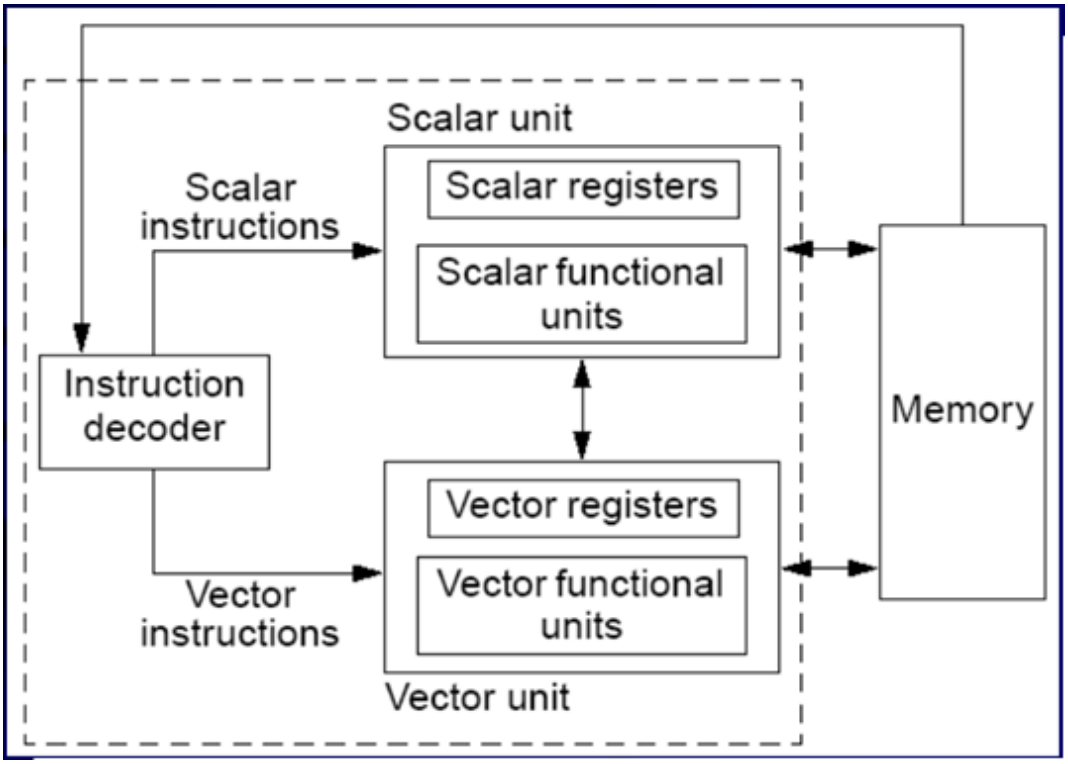


Figure 15 : l'architecture d'un processeur vectoriel [15].

Parmi les calculateurs vectoriels achevé, on peut citer : CDC Cyber 205, CRAY, IBM 3090, NEC SX, Fujitsu VP, Hitachi S8000.

2.2 Approche de programmation

Le paradigme de programmation Multithreading est le plus utilisé dans les architectures SIMD.

2.2.1 Paradigme de programmation Multithreading

La programmation parallèle la plus simple consiste à avoir plusieurs threads ou flots d'instruction qui s'exécutent en manipulant des données stockées dans une mémoire commune. Elle est donc naturellement le type de programmation le plus employé sur les architectures à mémoire partagées. Un thread est défini comme étant une courte séquence d'instructions pouvant être exécutée en tant qu'une seule unité par un processeur [10].

Le programmeur peut utiliser dans ce cas des outils spécialisés comme les bibliothèques implantant la norme POSIX à l'aide d'un langage qui intègre les threads dans sa sémantique, comme le C et le Java. La difficulté de ce modèle de programmation est la synchronisation des accès à la mémoire entre les tâches concurrentes. Aussi, pour

améliorer les performances, il suffit d'avoir toujours suffisamment de threads concurrents à exécuter pour occuper au maximum les différents processeurs.

Le Multithreading peut être vu comme un parallélisme d'instructions. Il consiste donc à décomposer un programme en plusieurs tâches (threads) qui peuvent alors être exécutées simultanément. Ce type de threads est souvent appelé des threads utilisateurs.

Le principal avantage du Multithreading est que cette solution est utilisable sur des architectures à mémoire partagée ce qui améliore efficacement les performances particulièrement en termes de temps de communication.

Cependant, l'inconvénient majeur dans ce cas est le surcoût relatif au basculement entre threads. En effet, on doit enregistrer l'environnement d'exécution du thread qu'on veut mettre en attente et chargé les registres du processeur avec le contexte d'exécution du nouveau thread à exécuter.

2.2.2 Discussion

Dans une architecture SIMD, chaque traitement est exécuté sur un ensemble différent de données par unité de traitement différente. De tels processeurs sont hautement spécialisés pour les problèmes numériques exprimés sous forme de matrice ou de vecteur, c'est à dire ils opèrent sur des vecteurs de données d'où le nom de processeurs vectoriels. Fonctionnement général des unités SIMD concerne :

- Chargement des données de la mémoire vers les registres SIMD
- Opération sur les registres SIMD
- Placement du résultat en mémoire

Dans ce cas et du point de vue du programmeur, cela signifie qu'il peut utiliser des instructions sur des vecteurs dans ses programmes. Sachant que le compilateur traduit ces instructions en instructions sur des vecteurs au niveau machine c'est à dire le compilateur s'occupe de « vectoriser » le code automatiquement.

Cependant, le programmeur est obligé parfois à utiliser des langages de bas-niveau (programmation bas-niveau) et à manipuler directement les registres et les instructions. Comme il peut des bibliothèques de haut-niveau pour le masquage de la programmation vectorielle.

La communication entre processeurs est prise en charge par le programmeur en utilisant un mécanisme d'échange de messages qui sert à traduire les transferts d'informations et les nécessaires synchronisations entre processeurs. Il existe plusieurs outils standards à utiliser pour cela, tels que: MPI, PVM, etc.

2.3 Exemple illustratif |9]

Soit l'ensemble de 16 nombres, les étapes du calcul de l'addition de ces nombres sur 4 processeurs en utilisant une architecture SIMD sont les suivants [9]:

Étape 1: l'unité de contrôle divise les nombres en 4 sous-ensembles, un seul ensemble par processeur (Figure16).

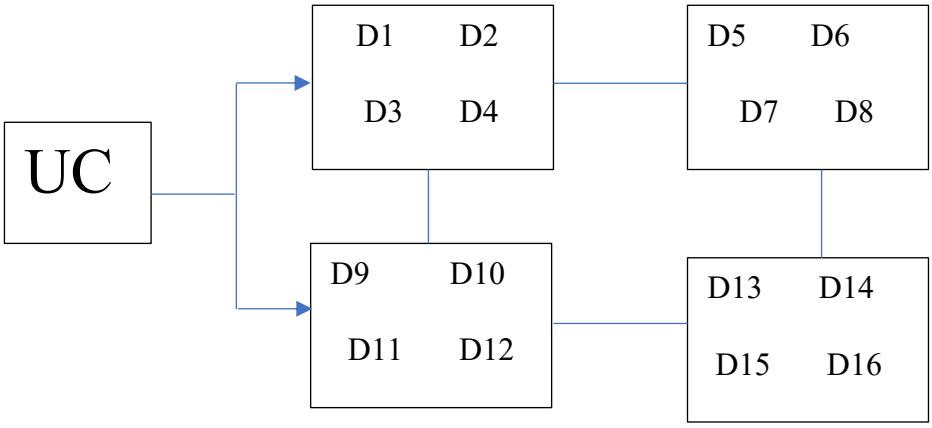


Figure 16: L'étape 1 de l'addition de 16 nombres sur 4 processeurs SIMD.

Étape 2: calcul des sommes partielles sur les processeurs individuels; calcul sériel sur chaque unité (Figure 17).

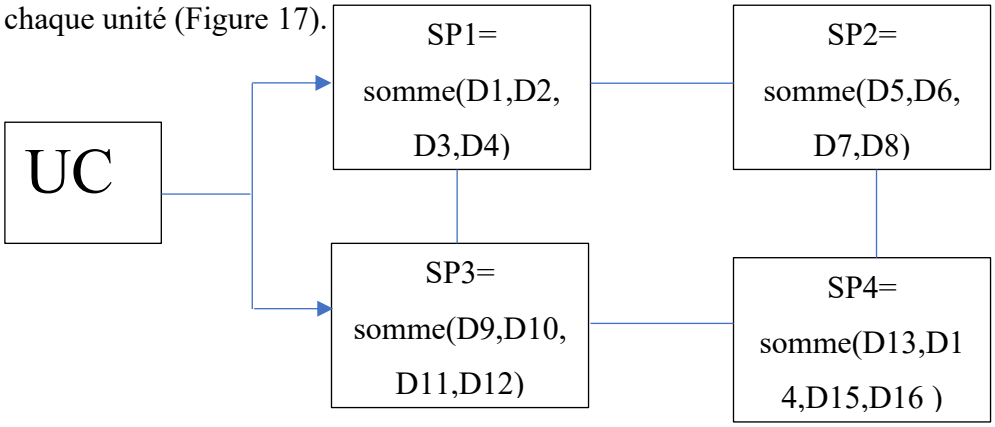


Figure 17: L'étape 2 de l'addition de 16 nombres sur 4 processeurs SIMD.

Étape 3: La moitié des processeurs envoie les données, l'autre moitié les reçoit et effectue une addition. Le réseau d'interconnexion est utilisé pour communiquer les données (Figure18).

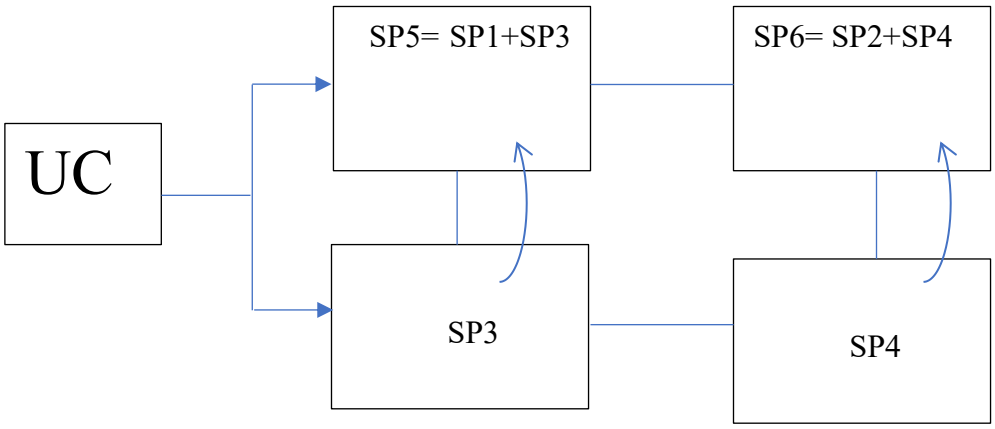


Figure 18: L'étape 3 de l'addition de 16 nombres sur 4 processeurs SIMD.

Étape 4: Additionner à nouveau les sommes partielles. Le résultat final est donc obtenu (Figure 19).

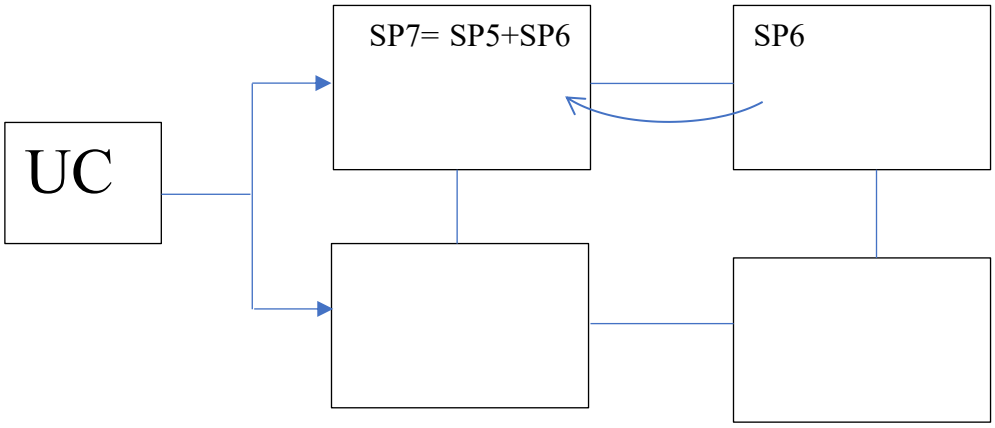


Figure 19: L'étape 4 de l'addition de 16 nombres sur 4 processeurs SIMD.

Le programme d'addition SIMD est le suivant:

/ calcul des sommes initiales partielles*

For (i = 0; i < 4; i = i + 1)

somme = somme + vecteur_local [i];

/ diviser pour régner pour obtenir le résultat final*

Limite = 4;

Moitié = 4

Repeat

moitié = moitié / 2;

if (Pn >= moitié && Pn < limite)

send (Pn – moitié, somme); / envoyer la somme à Pn/2 à travers le réseau*

limite = moitié;

Until (limite == 1);

Programme 1: l'addition de 16 nombres sur 4 processeurs SIMD.

3. Architecture MISD

Dans une architecture MISD, chaque processeur exécute une séquence différente d'instructions sur les mêmes données utilisées par les autres processeurs.

3.1 Aspects architecturaux

Dans une architecture MISD, plusieurs flux d'instructions sont appliqués sur un seul flux de données (Figure 20), et par conséquent une seule séquence de données est transmise à un ensemble de processeurs. Théoriquement, certains auteurs considèrent que cette classe n'est pas commercialisé et d'autres auteurs considèrent le pipeline comme un schéma MISD [9]. Par conséquent, l'architecture MISD reste un modèle théorique car il n'existe aucune machine construite sur ce modèle [4]. Alors que d'autres chercheurs considèrent les GPU (Graphical Processing Unit) comme une architecture MISD [13].

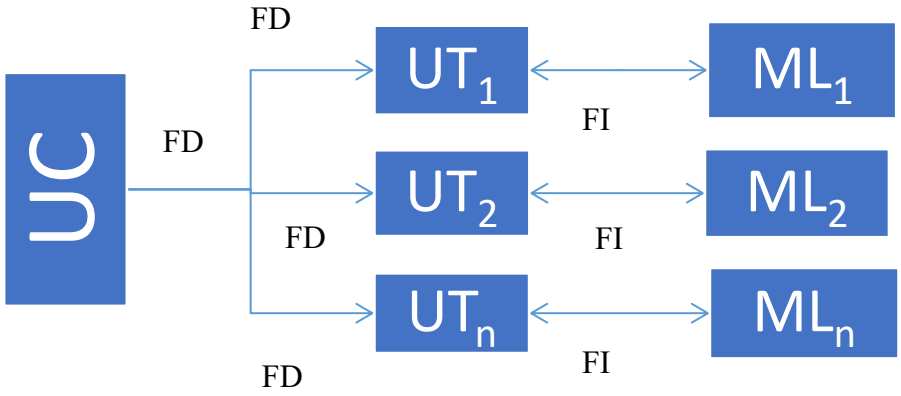


Figure 20 : L'architecture MISD.

3.2 Approche de programmation

3.2.1 Le paradigme message-passing

Ce modèle de programmation est naturellement adéquat aux architectures parallèles dans lesquels chaque processeur possède son propre espace mémoire [10].

Dans le paradigme de programmation message-passing, les programmeurs voient leurs programmes comme une collection de processus coopérants avec des variables privées (locales) [10]. Il faut exprimer dans son programme explicitement quelles vont être les données transmises et à quels moments les transmissions doivent se faire. Cependant, l'affectation des processus aux processeurs ainsi que des identificateurs des processus est pris en charge par l'environnement et non pas le programmeur.

La communication entre processus et l'échange de données se fait avec deux primitives d'échange de messages, la primitive d'envoi « send » et la primitive de réception « receive ». Il existe deux modes de communication le mode bloquant et le mode non-bloquant appelés aussi mode synchrone et asynchrone [10]. Ces modes sont introduits pour augmenter les performances du système.

Dans le cas d'un « send » bloquant, le processus envoie le message à sa destination et attend que le récepteur le reçoive avant de poursuivre son traitement. Cependant, Dans

le cas d'un « send » non-bloquant, le processus envoie le message et poursuit son traitement sans attendre [10].

Le format général de chaque primitive est le suivant :

send (adresse de la destination , nom de la variable, taille) ;

receive (adresse de l'émetteur , nom de la variable, taille) ;

En plus de ces deux primitives principales, les modèles de message-passing fournissent d'autres types de communication tels que les communications collectives qui permettent, par exemple, pour envoyer un message à tous les processus d'utiliser la primitive « broadcast » (diffusion général) et pour recevoir les résultats d'un calcul de tous les processus au niveau d'un processus, d'utiliser la primitive « Gather ».

Néanmoins, les communications sur des machines à mémoire distribuée sont toujours assez coûteuses en temps par rapport à leur puissance de calcul. Cela induit que la programmation parallèle devient plus complexe : il faut essayer de minimiser les communications, dans le même temps, que de partager les données et les activités pour maintenir les processeurs occupés.

3.2.2 Discussion

Dans ce mode, la même séquence de données est traitée par plusieurs processeurs en parallèle. Du point de vue du programmeur, cela signifie que chaque processeur va travailler sur une copie de cette séquence de données enregistrée dans sa mémoire locale. Cela augmente le risque de l'incohérence de données. C'est pourquoi, il est nécessaire que le maintien de la cohérence doive être contrôlé par le hardware, le software et le programmeur lui-même.

La communication entre processeurs est prise en charge par le programmeur en utilisant un mécanisme d'échange de messages qui sert à traduire les transferts d'informations et les nécessaires synchronisations entre processeurs. Il existe plusieurs outils standards à utiliser pour cela, tels que: MPI, PVM, etc.

Cette architecture peut être utilisée dans le filtrage numérique et la vérification de la redondance dans les systèmes critiques.

3.3 Exemple illustratif [9]

Soit l'ensemble de 16 nombres, les étapes du calcul de l'addition, la soustraction, le produit et la division de ces nombres en utilisant une architecture MISD sont les suivants :

Etape 1 : l'unité de contrôle envoie les mêmes 4 nombres et des flux d'instruction différents vers tous les processeurs (Figure 21).

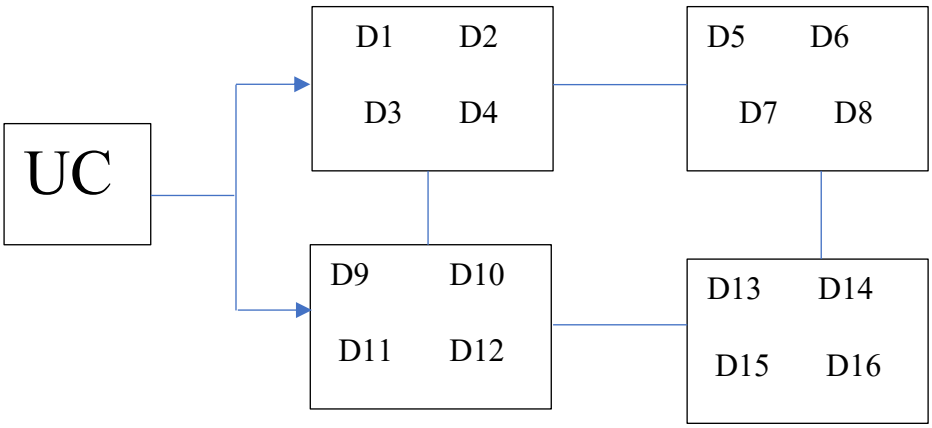


Figure 21 : L'étape 1 de la somme, soustraction, produit et la division de ces nombressur 4 processeurs MISD.

Etape2 : calcul des résultats partielles sur les processeurs individuels (Figure 22).

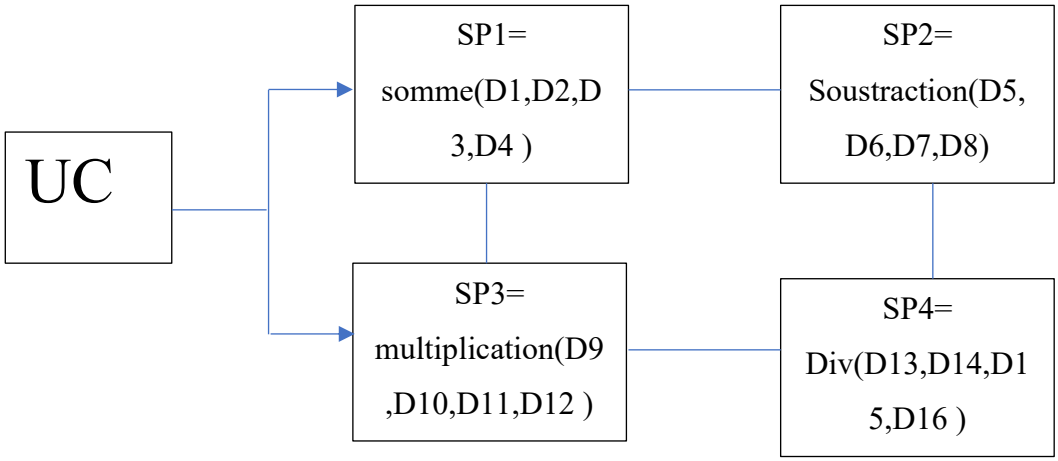


Figure 22 : L'étape 2 de la somme, soustraction, produit et la division de ces nombressur 4 processeurs MISD.

Etape 3 : Les résultats partiels sont donc obtenus (Figure 23).

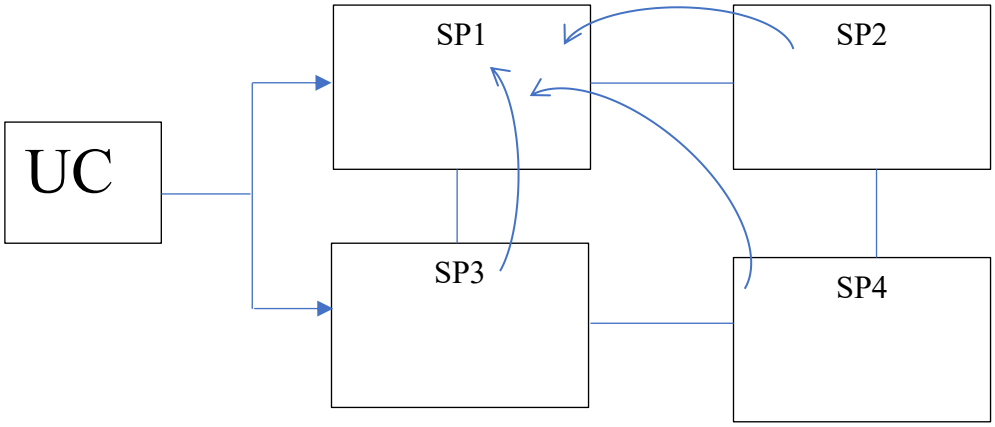


Figure 23 : L'étape 3 de la somme, soustraction, produit et la division de ces nombressur 4 processeurs MISD.

Dans ce cas, le programme MISD est le suivant:

```
/* calcul des sommes

Somme=0 ;

For (i = 0; i < 4; i = i + 1)

somme = somme + vecteur_local [i];    /* vecteur_local [i] contient Di

if (Pn =! 1)

send (P1, somme); /* envoyer la somme à P1 à travers le réseau

/* calcul de soustraction

sous = vecteur_local [1] ;

For (i = 1; i < 4; i = i + 1)

sous = sous -vecteur_local [i];

if (Pn =! 1)

send (P1, sous); /* envoyer la somme à P1 à travers le réseau

/* calcul du produit
```

```
prod = 1 ;
For (i = 0; i < 4; i = i + 1)
prod = prod*  vecteur_local [i];
if (Pn =! 1)
send (P1, prod); /* envoyer la somme à P1 à travers le réseau
/* calcul de division
div = vecteur_local [i];
For (i = 1; i < 4; i = i + 1)
div = div/  vecteur_local [i];
if (Pn =! 1)
send (P1, div); /* envoyer la somme à P1 à travers le réseau
```

Programme 2 : calcul de la somme, soustraction, produit et la division sur 4 processeurs MISD.

4. Architecture MIMD

L'architecture MIMD est une architecture dont les unités de traitement exécutent des différents flux d'instructions qui opèrent sur des flux de données différents.

4.1 Aspects architecturaux

L'idée derrière le MIMD est de combiner de petits calculateurs pour obtenir un calculateur puissant dont le but est de répandre la disponibilité de petits microprocesseurs de haute performance à un coût réduit (Figure 24). Notons que, le MIMD permet d'assurer une plus grande fiabilité par rapport à la disponibilité et améliore la stabilité [9].

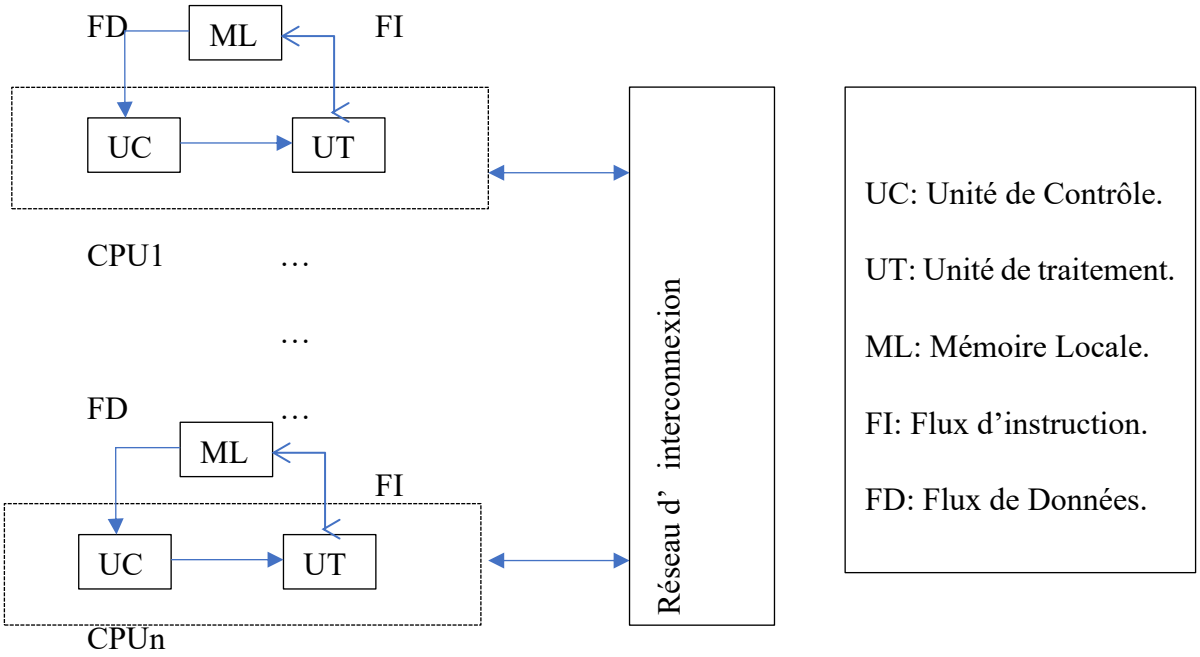


Figure 24: l'architecture MIMD.

Le MIMD est un ensemble de processeurs à usage général exécutent, simultanément, différentes séquences d'instructions sur différents flux de données [9]. Chaque processeur traite une donnée distincte, et lui applique son propre flux d'instructions. Chaque processeur est caractérisé par :

- Une petite taille.
- Un prix réduit.
- Des hautes performances.

4.1.1 Classification des architectures MIMD :

Les architectures MIMD peuvent être classées selon plusieurs critères, tels que :

- L'organisation de la mémoire
- La topologie du réseau d'interconnexion.

4.1.1.1 L'organisation de la mémoire

Les processeurs MIMD peuvent utiliser soit [1]:

- Une mémoire partagée (Shared Memory : SM) : tous les processeurs accèdent à la totalité de la mémoire.

- Une mémoire distribuée (Distributed Memory : DM) : chaque processeur dispose d'une mémoire locale .

Sachant que, il est possible de concevoir une architecture MIMD à mémoire hybride (Hybrid Memory). Il s'agit d'une architecture à mémoire partagée et mémoire distribuée à la fois.

MIMD à mémoire partagée [9] : Dans une architecture MIMD à mémoire partagée, un espace mémoire global est "visible" par tous les processeurs (Figure 25). En plus, les processeurs peuvent avoir une mémoire locale dans laquelle sera copiée une partie de la mémoire globale pour des raisons d'optimisation des accès. L'un des exemples de processeurs MIMDs à mémoire partagée est les processeurs SMP (Symmetric Multi-Processors).

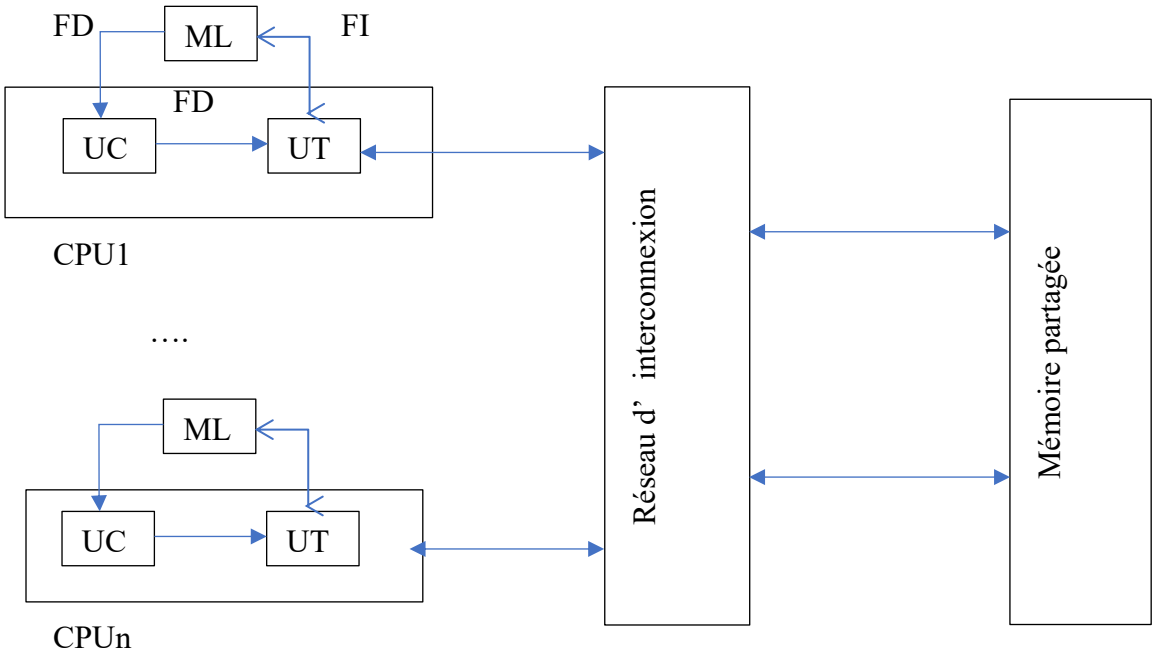


Figure 25: MIMD à mémoire partagée

Les avantages de l'architecture MIMD sont :

- Un espace global adressable qui facilite la programmation parallèle.
- Le partage des données entre les tâches est rapide.

Les inconvénients sont les suivants :

- Ce modèle ne favorise pas la scalabilité :
  - Ajouter un CPU augmente le trafic au travers la mémoire partagée.

- Problème de cohérence des données :
    - La cohérence entre les copies locales et la mémoire globale doit être gérée à la fois par le hardware, le software et parfois même par l'utilisateur.

Les processeurs SMP (Symmetric Multi-Processors) : Un processeur SMP est un système à mémoire partagée, lorsqu'un processeur lit ou écrit une donnée en mémoire, cette donnée est transférée dans le cache du processeur. Cependant, des mécanismes de cohérence des données sont nécessaire. Le fait que, tous les processeurs utilisent le même bus mémoire, limite la scalabilité de l'architecture.

Lorsque le nombre de processeurs rapides augmente, la contention mémoire peut dégrader sérieusement les performances ; ces architectures ne supportent pas un grand nombre de processeurs (16 à 64).

MIMD à mémoire distribuée [9] :

Les principes de fonctionnement de ce type d'architecture sont les suivants :

- Un espace mémoire différent est associé à chaque processeur (Figure 26).
- Du point de vue du programmeur, il n'y a aucune variable partagée : une variable ne peut être accédée que par un processeur unique.

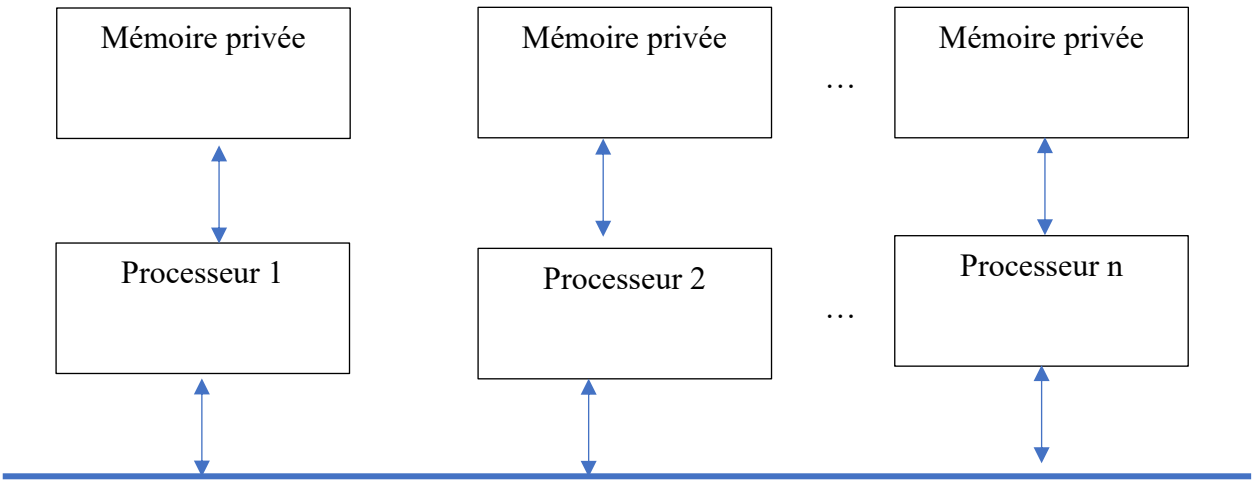


Figure 26: MIMD à mémoire distribuée

En plus, l'accès à la mémoire d'un autre processeur se fait par envoi/réception de messages à travers le réseau d'interconnexion (Figure 27). Pour la communication, le programmeur utilise des canaux de communication et des primitives « send » et «

receive ». Donc, il n'y a aucune compétition entre les processeurs pour une mémoire partagée :

- Le nombre de processeurs n'est pas limité par la contention mémoire ;
- La vitesse du réseau d'interconnexion est un paramètre important pour les performances globales ;
- Les algorithmes utilisés doivent minimiser les échanges de données, en distribuant les données sur les mémoires locales.

Un exemple d'architecture MIMD à mémoire distribuée est les Clusters.

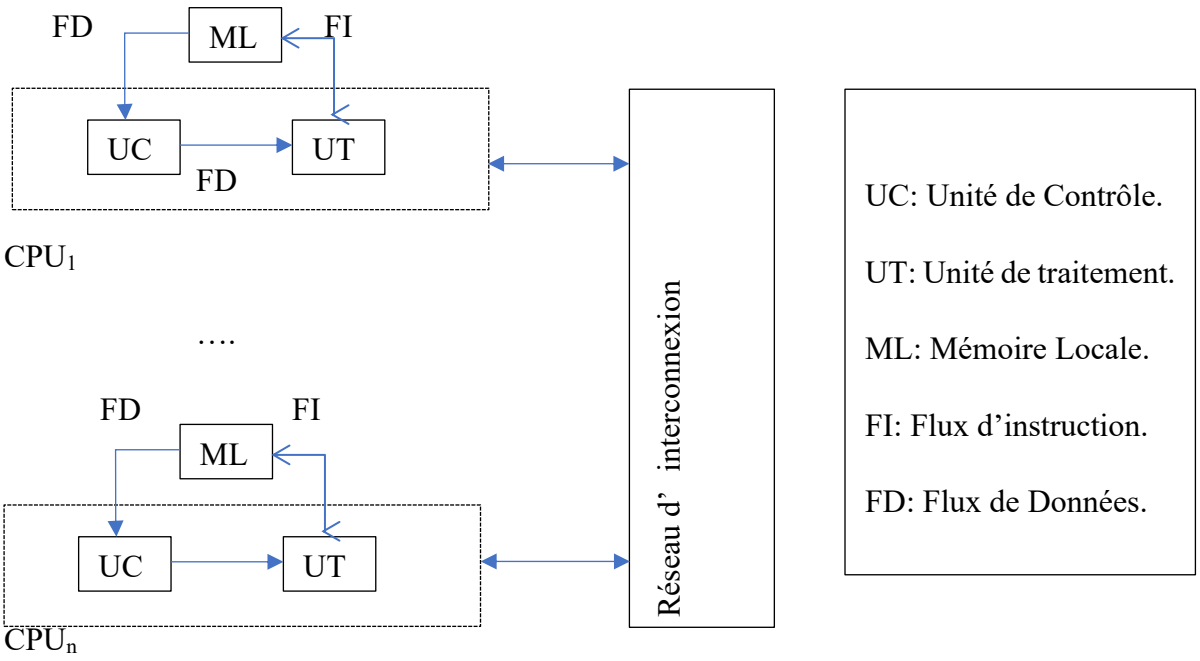


Figure 27: Le réseaux d'interconnexion dans une architecture MIMD à mémoire distribué.

Les avantages de ces architectures sont :

- La scalabilité est facilitée (l'ajout d'un processeur n'influe pas sur le fonctionnement des autres).
- Chaque processeur a accès à sa propre mémoire sans interférence ni problèmes de cohérence de cache.
- Utilisation souvent plus optimale des mémoires à accès rapide du processeur (cache).

Néanmoins, cette architecture peut avoir quelques inconvénients tels que :

- Le programmeur doit gérer lui-même les échanges de données entre processeurs.
- La mise en œuvre est parfois difficile et elle nécessite souvent de nouveaux algorithmes de distribution des données cohérentes.
- Le surcoût dû aux communications peut dans certains cas faire chuter considérablement les performances.

Les Clusters : ils sont des groupes de calculateurs complets interconnectés. Les calculateurs individuels travaillent ensemble comme une ressource de calcul unique. Cependant, chaque calculateur autonome dans un cluster est nommé « nœud ». Les processeurs NUMA sont des clusters.

Les avantages des clusters sont :

- ***Scalabilité absolue :*** il est possible de créer de gros clusters qui dépassent de loin la puissance des machines individuelles les plus puissantes.
- ***Scalabilité incrémentale :*** un cluster est configuré de telle sorte qu'il soit possible d'ajouter de nouveaux nœuds au système par petits incréments.
- ***Haute disponibilité :*** du fait que chaque nœud d'un cluster soit un calculateur individuel, une panne sur un nœud n'entraîne pas de perte de service. La tolérance aux fautes est souvent gérée de manière automatique par le logiciel.
- ***Meilleur rapport performance et prix :*** en utilisant les possibilités de construction de blocs, il est possible d'obtenir un cluster ayant une puissance de calcul plus grande qu'une grosse machine, à un prix réduit.

Il existe différentes configurations de clusters tels que (1) la configuration à lien direct et (2) la configuration avec partage de disque. La figure 28 illustre le cas d'un cluster à liens directs sans disques partagés et la Figure 29 le cas d'un cluster avec disques partagés.

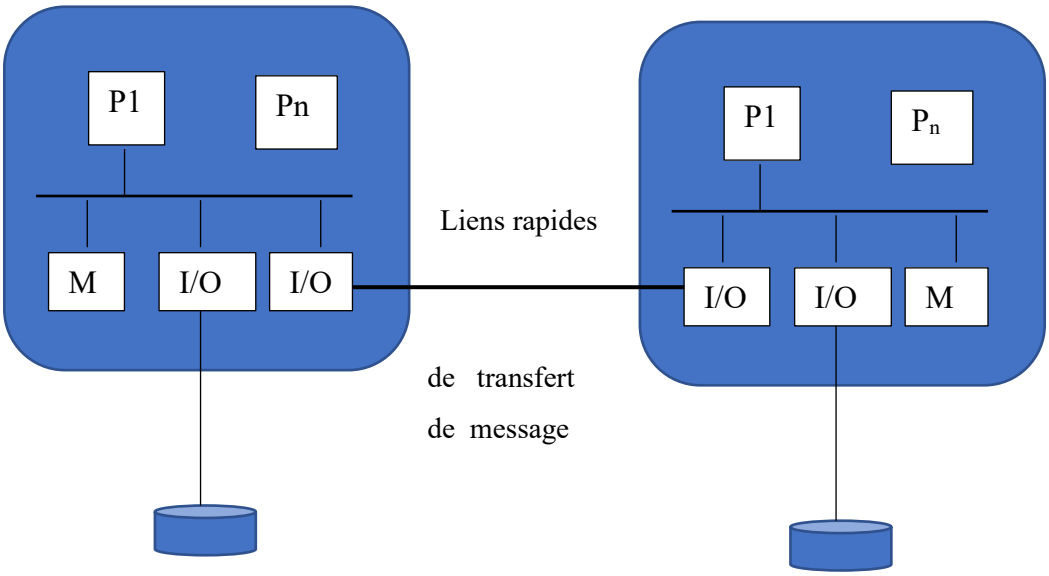


Figure 28: Clusters à liens directs sans disques partagés

Clusters avec disques partagés (Figure 29) :

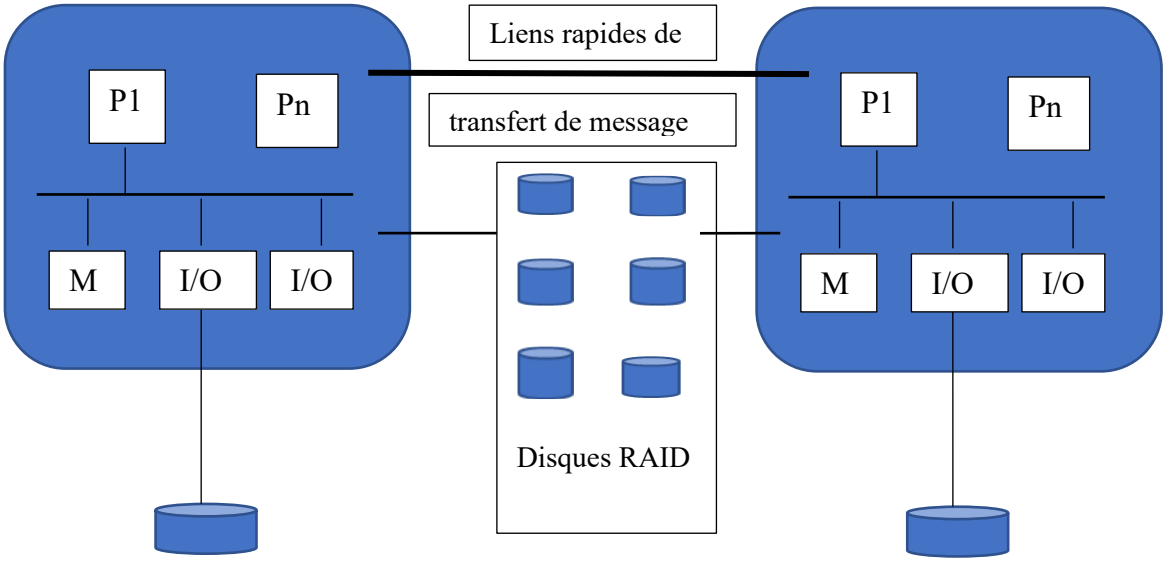


Figure 29: Clusters avec disques partagés

Les systèmes NUMA Non-Uniform Memory Access)

Dans une architecture UMA (Uniform Memory Access), tous les processeurs ont accès à toutes les parties de la mémoire principale. Le temps d'accès à la mémoire d'un processeur à toutes les régions de la mémoire est le même. En plus, les temps d'accès des différents processeurs est le même. Le SMP est un processeur UMA (Figure 30).

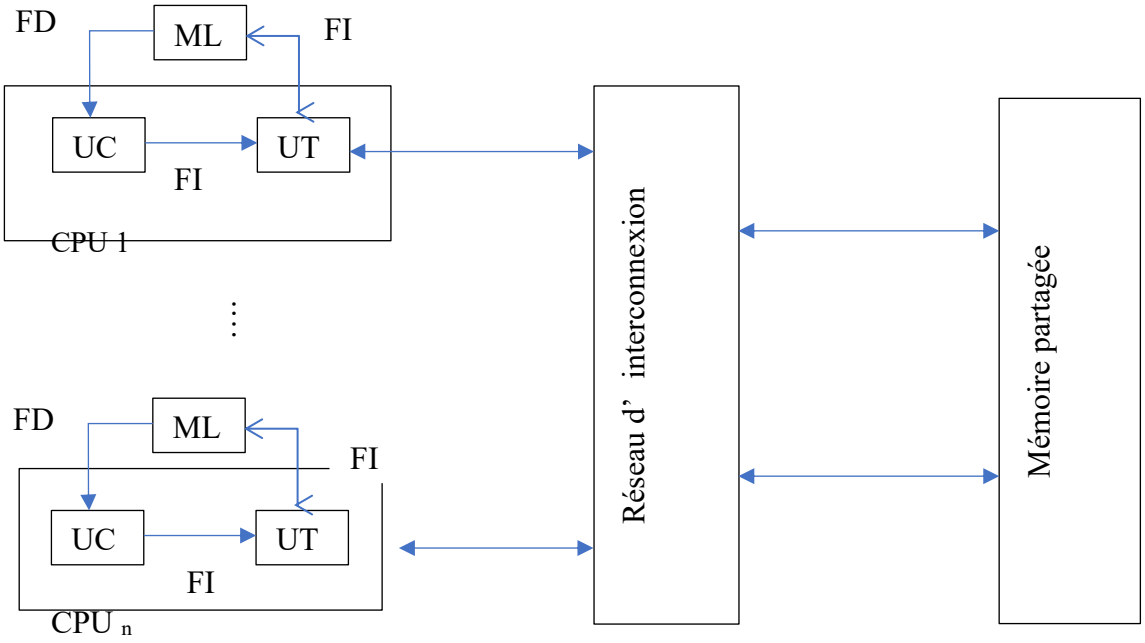


Figure 30: architecture d'un processeur SMP.

Cependant, NUMA tous les processeurs ont accès à toutes les parties de la mémoire principale. Cependant, le temps d'accès mémoire d'un processeur diffère selon la région de la mémoire à laquelle il accède, cela est vrai pour tous les processeurs. Aussi, les temps d'accès aux différentes régions diffèrent d'un processeur à un autre. Le système NUMA est conçu pour pallier les limites de l'architecture SMP qui présentent des performances limitées dès que le nombre de processeurs dépasse 64.

La Figure 31 illustre un exemple de processeur NUMA.

Un système **CC-NUMA (Cache-Coherent Non-Uniform Memory Access)** est un système NUMA dans lequel une cohérence de données est maintenue entre les caches des différents processeurs. Un système NUMA sans cohérence est équivalent à un cluster qui disposent de mémoires distribuées et assurent la cohérence par logiciel. Les systèmes CC-NUMA pour résoudre ce problème et garantir la cohérence des données en considérant la mémoire comme un espace mémoire global adressable par chaque processeur.

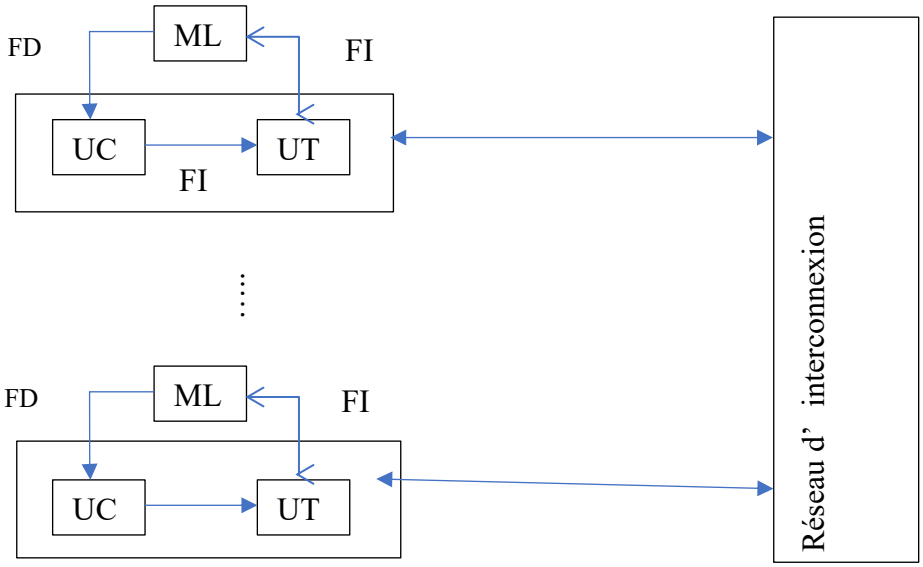


Figure 31:architecture d'un processeur NUMA.

4.1.1.2 Le mode de connectivité

Une deuxième classification est basée sur le mode de connectivité des différents processeurs. On obtient donc:

- Les architectures MIMDs à bus unique;
- Les architectures MIMDs en réseau.

MIMD à bus: Un bus unique est considéré comme la solution la plus simple de connecter des systèmes multiprocesseurs [16]. La figure 32 présente une illustration d'un système à bus unique. Dans sa forme générale, un tel système se compose de N processeurs, chacun ayant son propre cache, reliés par un bus de données unique et une mémoire principale partagée. Ils disposent de caches locaux pour réduire le trafic sur le bus. Par conséquent, un mécanisme de cohérence sera nécessaire dans ce cas.

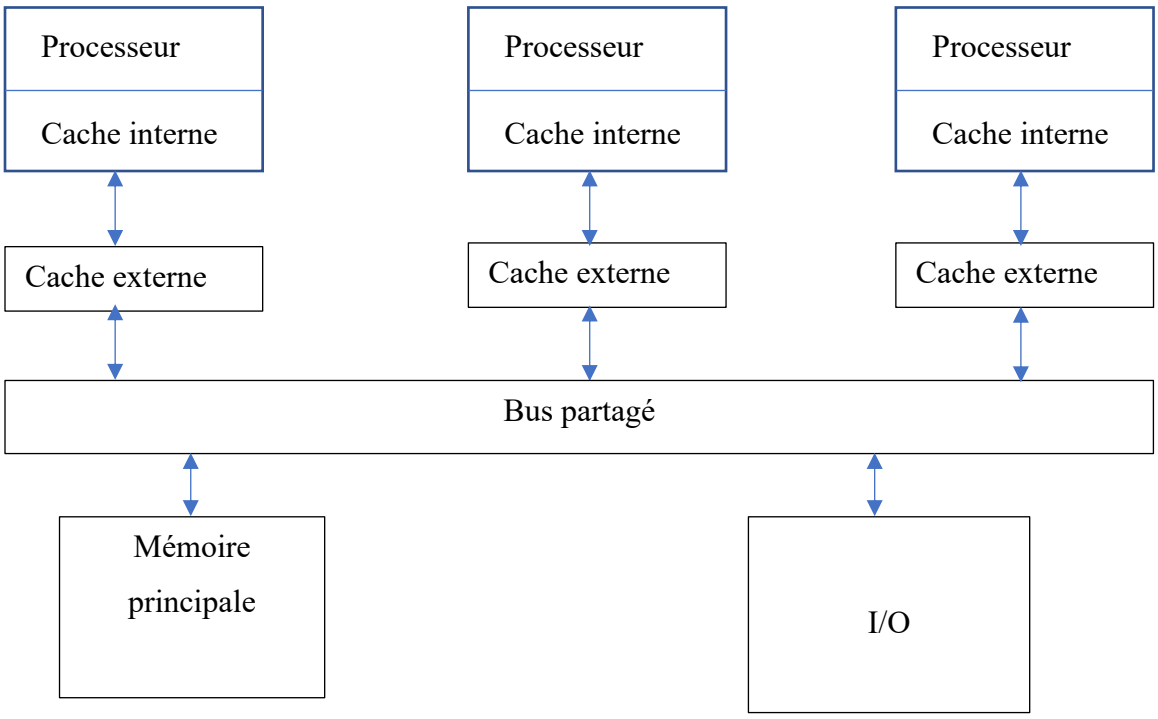


Figure 32 : MIMD à bus.

Les avantages de l'architecture MIMD à bus sont :

Simplicité: approche la plus simple parmi les organisations multiprocesseurs;

Flexibilité: le système est généralement facile à étendre en attachant de nouveaux processeurs au bus;

Fiabilité: le bus est essentiellement un médium passif, et une défaillance sur un nœud ne devrait pas entraîner de défaillance du système complet.

Le principal inconvénient du MIMD à bus est la performance:

- Toutes les références mémoire passent par le bus commun;
- Donc le temps de cycle du bus limite la vitesse du système;
- Nécessité de caches.

La table 5 illustre quelques exemples de processeurs MIMD à bus :

Institution	Nom	Nombre processeurs	Bits processeur	FPUs	Taille de mémoire (MB)	Bande passante (MB/s)	Année
Sequent	symmetry	30	32	30	240	53	1988
Silicon Graphics	4/360	16	32	16	512	320	1990
Sun	4/640	4	32	4	768	320	1991

Table 5 : Processeurs MIMD à bus [9].

MIMD en réseau: dans une architecture MIMD en réseau, chaque processeur a sa mémoire locale et les processeurs connectés à travers une topologie réseau. Donc, il est nécessaire de synchroniser entre les différents processeurs (Figure 33).

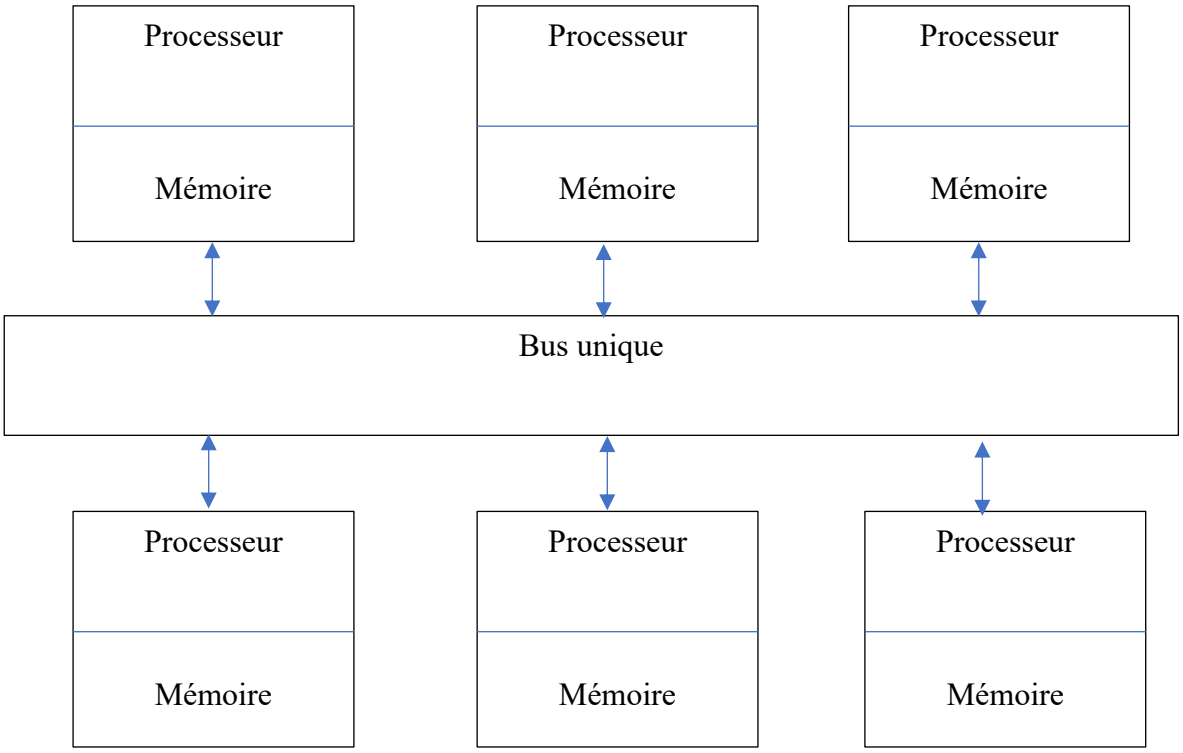


Figure 33 : MIMD en réseau.

La table 6 illustre quelques exemples de processeurs MIMD en réseau :

Institution	Nom	Nombre de processeurs	Horloge (MHz)	FPUs	Taille de mémoire (MB)	Bande passante (MB/s)	Année
Intel	iPSC/2	128	16	128	512	896	1988
nCube	ten	1024	10	1024	512	10240	1987
Intel	Delta	540	40	540	17280	21600	1991
Thinking machines	CM-5	1024	33	4096	32768	5120	1991

Table 6:Processeur MIMD en réseau [9].

4.1.2 Réseau d'interconnexion

Dans une architecture MIMD en réseau, le réseau d'interconnexion est un composant clé de l'architecture car il a une influence décisive sur les performances et le coût global du système. Le trafic dans le réseau consiste en des transferts de données, de commandes et de requêtes. Les paramètres clés du réseau sont:

- Bande passante totale mesurée en bits/seconde.
- Coût.

Il définit les connexions entre les nœuds (mémoire et processeurs), il existe deux types de réseaux :

- Réseaux statiques : Les réseaux statiques utilisent des liens (connections) directs (es) qui sont fixes une fois construits. Ce type de réseau est plus adapté à la construction d'ordinateurs où les schémas de communication sont prévisibles ou réalisables avec des connexions statiques [15].
- Réseaux dynamiques : il est convenable parfois d'utiliser des connexions dynamiques qui peuvent mettre en œuvre tous les modèles de communication en fonction des exigences du programme [15]. Il s'agit des réseaux configurables dynamiquement pour répondre à la demande de ce dernier.

Ils peuvent utiliser des connexions directes (réseaux directes) entre les différents nœuds ou des connexions indirectes (réseaux indirectes) [17]. Un réseau est dit directe si chaque nœud est directement connecté à ces voisins. Cependant, un réseau est dit indirect si tous les nœuds sont connectés indirectement les uns aux autres.

Le problème d'implémentation majeur concerne la longueur des liens qui affecte la fréquence d'horloge et les besoins en puissance.

Il existe plusieurs topologies d'interconnexion des processeurs, tels que [15] [9]:

1. **Topologie en bus unique (Figure 34):**

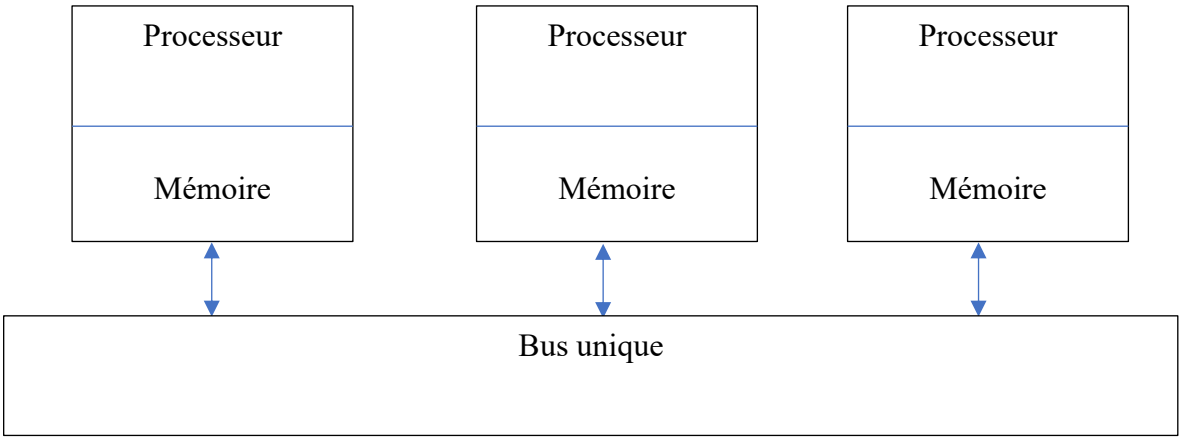


Figure 34 : Topologie en bus unique (cas MIMD distribué).

Cette topologie est caractérisée par :

- Les réseaux en bus sont simples et peu coûteux;
- Une seule communication est autorisée à la fois;
- La bande passante est partagée entre les différents nœuds;
- Les performances sont relativement mauvaises;
- Pour maintenir une certaine performance, le nombre de nœuds doit être limité (16 – 20).

2. **Topologie en réseau complètement connecté (Figure 35) :**

Dans un réseau complètement connecté, chaque nœud est connecté à tous les autres nœuds du réseau. Les réseaux entièrement connectés garantissent une livraison rapide des messages de n'importe quel nœud source à n'importe quel nœud de destination (un seul lien doit être traversé). Le routage des messages dans ce cas devient une tâche simple. Cependant, les réseaux complètement connecté sont coûteux en termes de nombre de liens nécessaires à leur construction (le cout dépend fortement du nombre des nœuds N) [17].

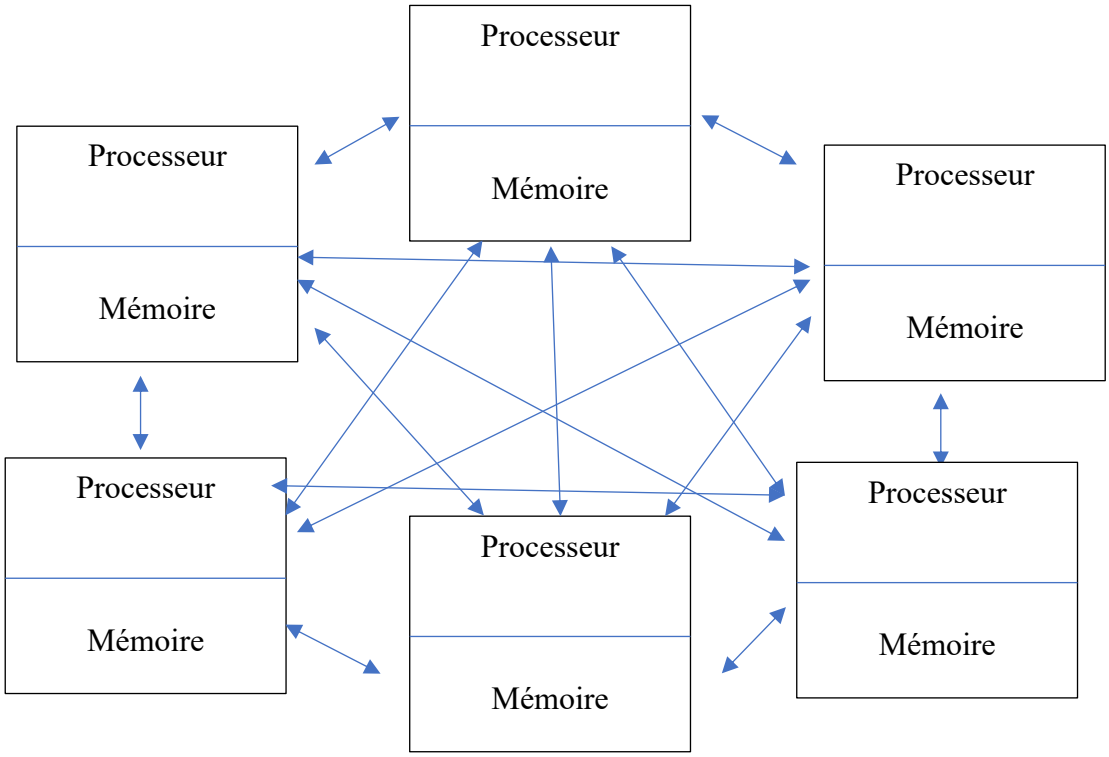


Figure 35 : Topologie en réseau complètement connecté.

Cette Topologie est caractérisée par :

- Chaque nœud est connecté à chacun des autres;
- Les communications peuvent être effectuées en parallèle entre n'importe quelle paire de nœuds;
- Les performances et le coût sont élevés;
- Le coût augmente rapidement avec le nombre de nœuds.

3. Topologie en crossbar (Figure 36):

La topologie en crossbar fournie les meilleures performances en termes de bande passante et du degré d'interconnexion [9]. Elle appartient à la famille des réseaux avec un seul étage [16] et elle offre une connexion dynamique entre chaque pair source-destination [15].

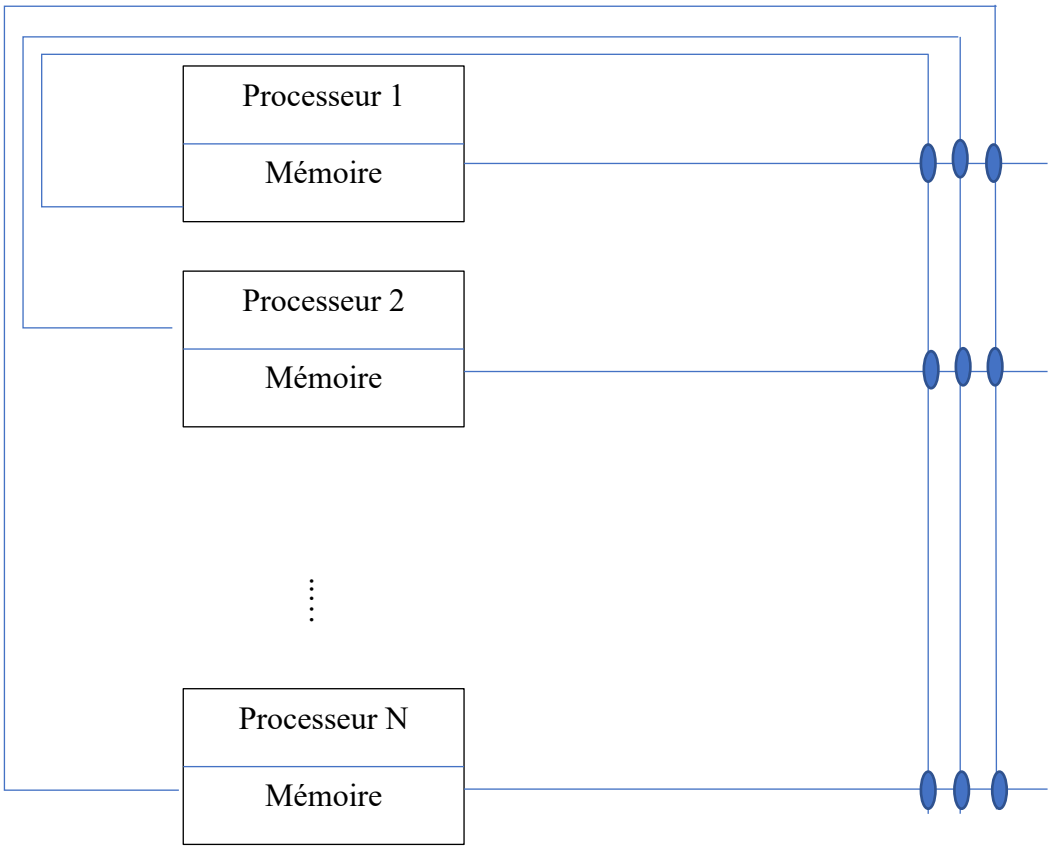


Figure 36 : topologie en crossbar.

Les caractéristiques de la topologie en crossbar sont :

- Le crossbar est un réseau dynamique: la topologie d'interconnexion peut être modifiée en positionnant les switchs;
- Le switch du crossbar est complètement connecté: n'importe quel nœud peut être directement connecté à n'importe quel autre;
- Moins d'interconnexions sont nécessaires que dans le cas du réseau statique complètement connecté;
- Un grand nombre de switchs est nécessaire;
- Un grand nombre de communications peuvent être effectuées en parallèle (un nœud donné peut recevoir ou envoyer seulement une donnée à la fois).

4. **Topologie en maillage (Figure 37):**

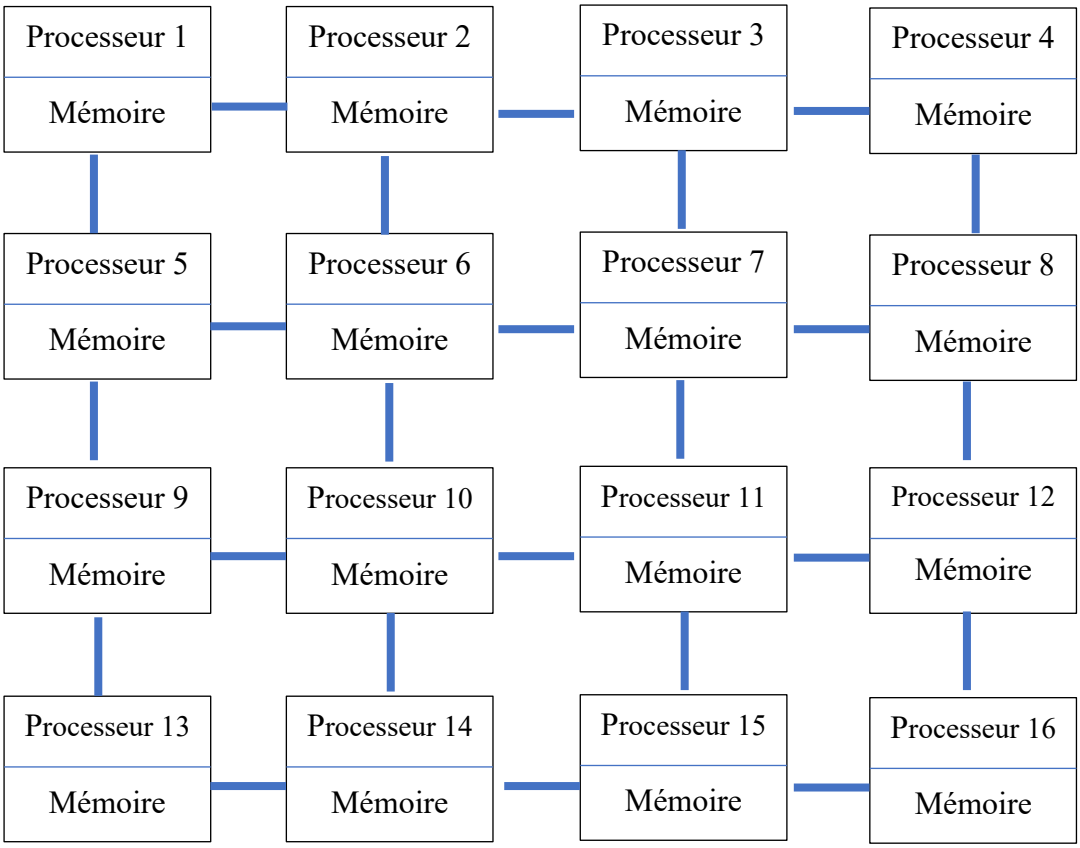


Figure 37 : Exemple de la topologie en maillage.

Les caractéristiques majeures de la topologie en maillage sont :

- Le réseau en maillage est moins coûteux que les réseaux complètement connectés;
- Il offre des performances relativement bonnes;
- Afin de transmettre des informations entre des nœuds donnés, un routage à travers des nœuds intermédiaires est nécessaire (au maximum 2 x (n-1) nœuds intermédiaires pour un maillage n x n);
- Il est possible d'offrir des connexions circulaires entre les nœuds 1 et 13, 1 et 4, 2 et 14 …);
- Des maillages 3D ont également été implémentés.

5. **Topologie en Topologie en hypercube (Figure 38):**

Il s'agit d'une architecture n-cube binaire qui a été mise en œuvre dans les systèmes iPSC, nCUBE et CM-2. En général, un n-cube est constitué de N= 2^n nœuds répartis sur *n* dimensions, avec deux nœuds par dimension [15].

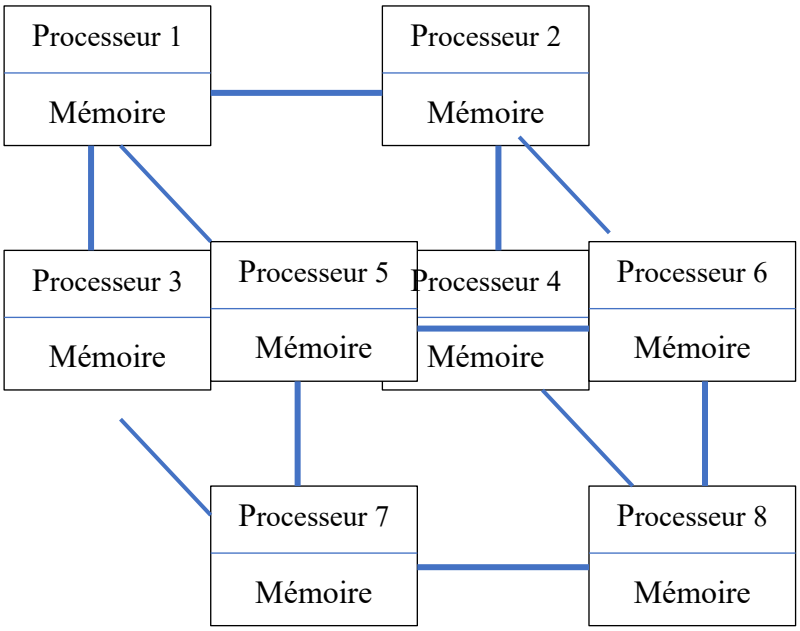


Figure 38 : Exemple de topologie avec un seul cube.

Cette topologie est caractérisée par le fait que :

- 2^n nœuds sont disposés dans un cube à *n* dimensions.
- Chaque nœud est connecté à *n* voisins.
- Afin de transmettre une information entre des nœuds donnés, un routage à travers des nœuds intermédiaires est nécessaire (maximum n intermédiaires).

Les métriques de performances de la topologie d'interconnexion concernent principalement :

- **Bande Passante totale du réseau:** est l'approximation du meilleur cas qui égale la bande passante d'un lien multipliée par le nombre de liens
- **Bande passante bi-section:** est une approximation du pire cas qui égale la somme des bandes passantes à travers un lien.

4.2 Approche de programmation

Tout d'abord, signalons que trop peu de programmes ont été écrits pour des processeurs parallèles à cause de:

- Le Surcoût en communication élevé.
- Ils requièrent la connaissance de la structure matérielle;
- Pas de portabilité entre les MIMD;
- Nouveaux algorithmes sont nécessaires pour tirer parti du parallélisme.

Les deux paradigmes de programmation parallèle Multithreading et Message-Passing peuvent être utilisés dans ce type d'architecture. Cependant le choix entre les deux est basé sur le type de la mémoire utilisé, c.-à-d. MIMD à mémoire partagée ou MIMD à mémoire distribuée.

Dans le cas général, le programmeur doit isoler les tâches de calcul, étudier leurs dépendances mutuelles et les synchroniser, gérer la communication, etc.

Le mot-clef dans ce cas est la localité des traitements et des données. Le programmeur doit chercher des algorithmes mettant en évidence des calculs sur des données locales en minimisant ainsi les transferts. Aussi, le programmeur doit avoir une idée claire sur le placement de ces deux quantités afin de savoir quelles tâches exécuter sur quels nœuds dans le but de minimiser le délai total de l'exécution du programme parallèle.

Un autre problème qui peut affecter les performances des programmes parallèles dans ce cas concerne l'équilibrage de la charge particulièrement en termes de calcul sur les différents nœuds en minimisant les communications. Le programmeur doit définir la taille de ses tâches en fonction du nombre de processeurs disponibles. Cependant, cela n'est pas toujours possible parce que les tailles de certaines tâches peuvent etre dynamiques.

Le mécanisme d'échange de messages sert à traduire les transferts d'informations et les nécessaires synchronisations entre processeurs. Il existe plusieurs outils standards à utiliser pour cela, tels que: MPI, PVM, etc.

4.3 Exemple illustrative [9]

4.3.1 **Programmation des MIMD en bus :** addition de 16 nombres sur 4 processeurs

Etape 1: chaque processeur calcule une somme partielle. Les nombres sont lus depuis la mémoire et les résultats sont stockés en mémoire (Figure 39).

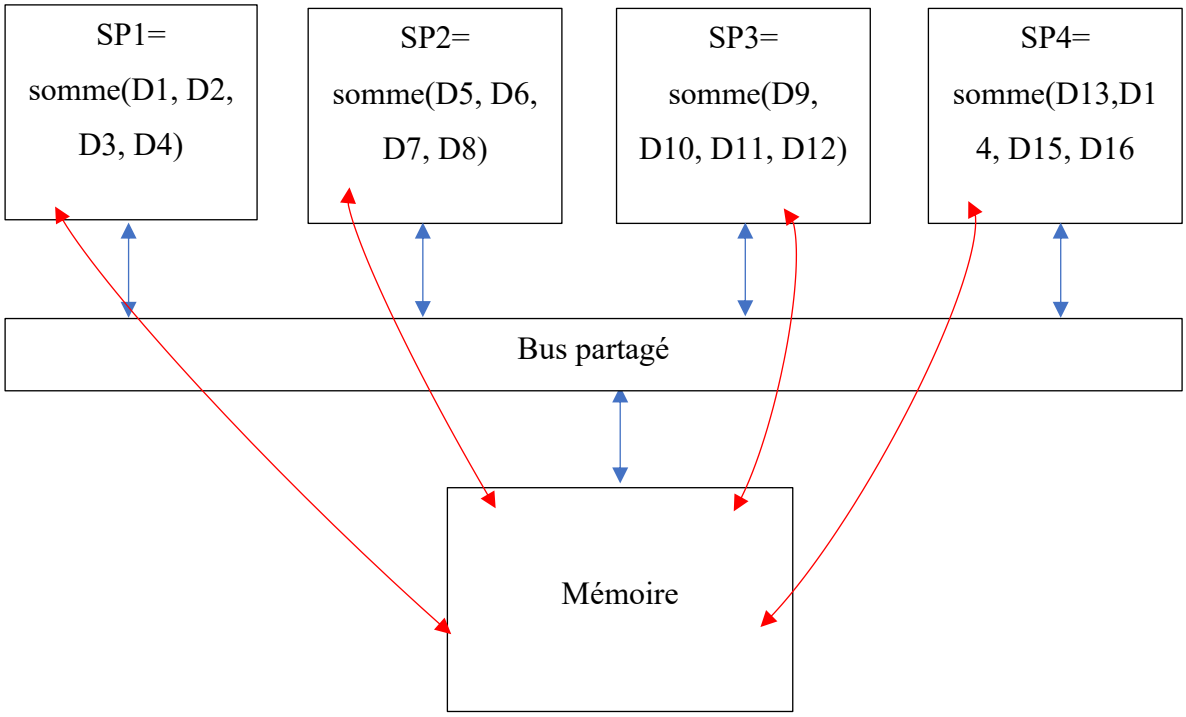


Figure 39: étape 1 de l'addition de 16 nombres sur 4 processeurs MIMD à bus unique.

Etape 2: addition des sommes partielles sachant que les sommes partielles sont lues en mémoire et les résultats sont stockés en mémoire (Figure 40).

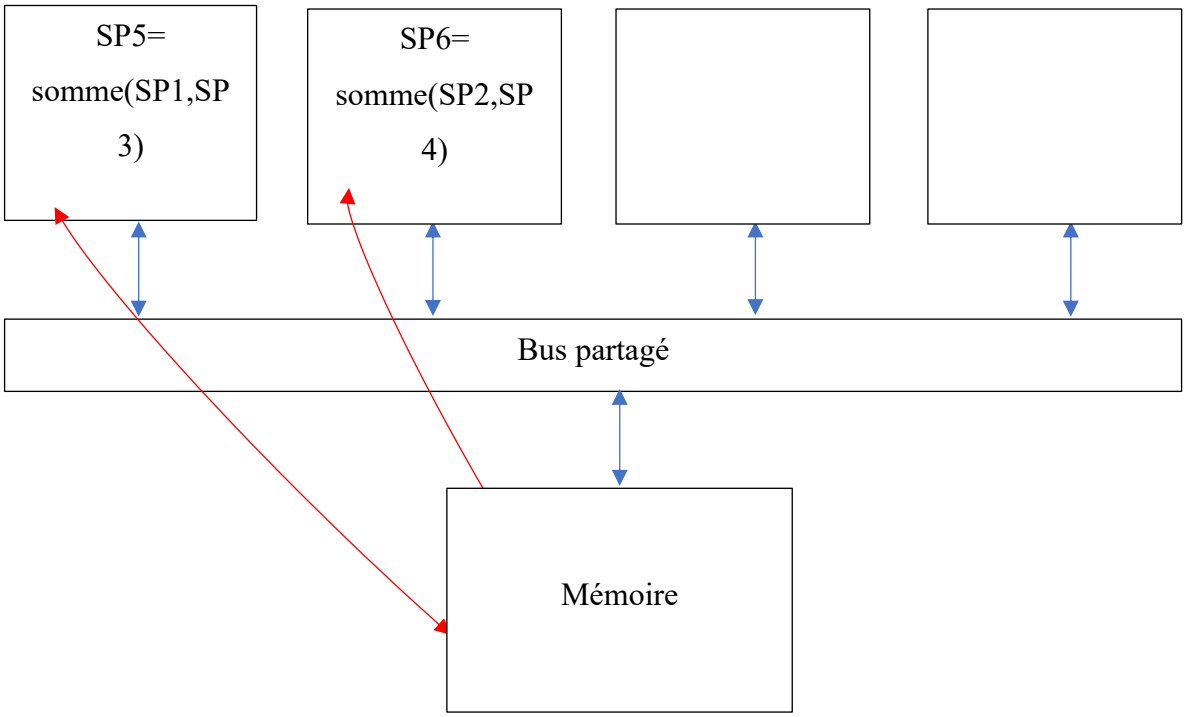


Figure 40 : étape 2 de l'addition de 16 nombres sur 4 processeurs MIMD à bus unique.

Etape 3: addition des sommes partielles, les sommes partielles sont lues en mémoire et le résultat final est stocké en mémoire (Figure 41).

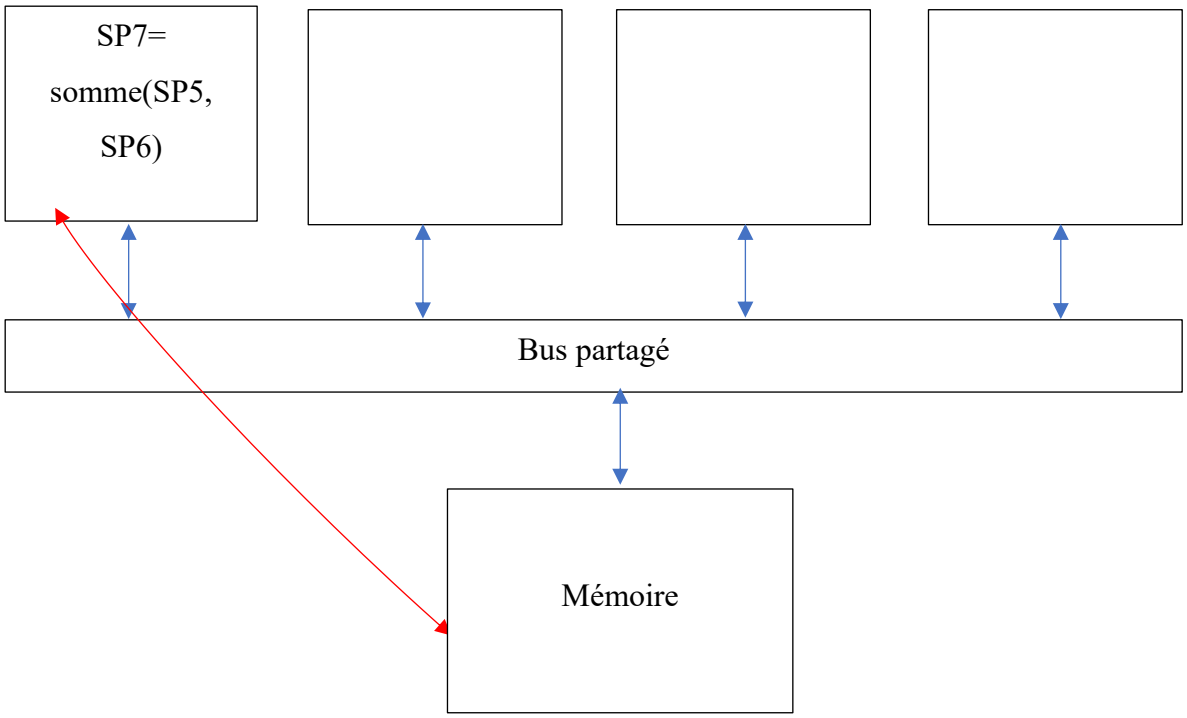


Figure 41 : étape 3 de l'addition de 16 nombres sur 4 processeurs MIMD à bus unique.

Le programme de l'addition dans ce cas est le suivant :

/ calcul des sommes initiales partielles*

Somme [Pn]

*For (i = 4*Pn; i < 4* (Pn + 1); i = i + 1)*

somme [Pn] = somme [Pn] + A [i];

/ diviser pour régner pour obtenir le résultat final*

Moitié = 4

Repeat

sync() / attendre la fin des sommes partielles*

moitié = moitié / 2;

if (Pn < moitié)

somme [Pn] = somme [Pn + moitié];

Until (moitié == 1);

Programme 3 : d'addition MIMD bus unique.

4.3.2 Programmation des MIMD en réseau : addition de 16 nombres sur 4 processeurs

Etape 1: distribution des nombres en 4 sous-ensembles, un par processeur (Figure 42).

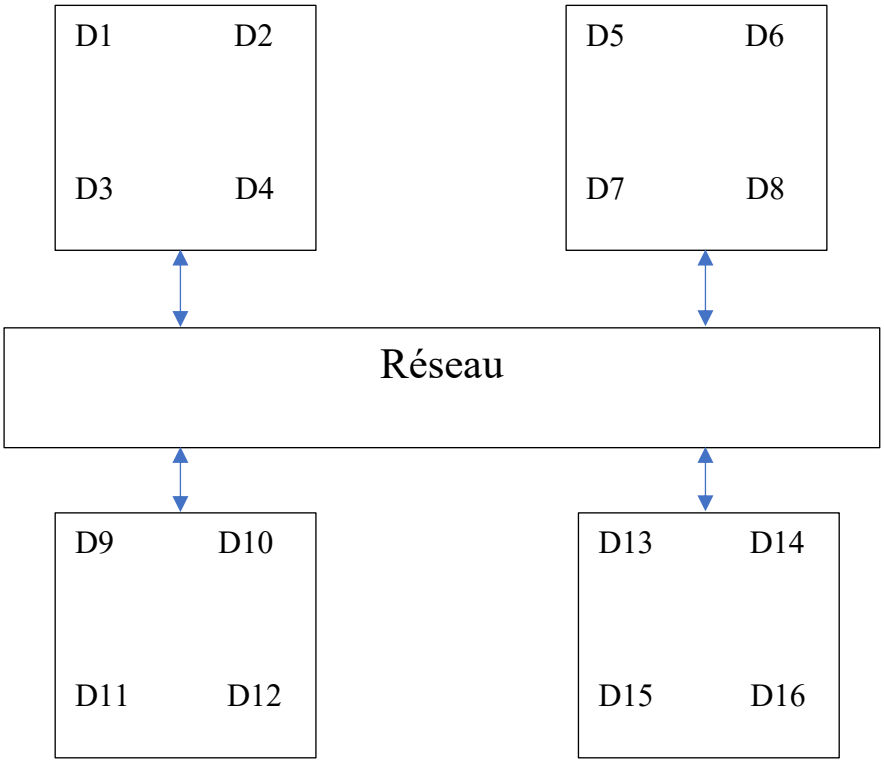


Figure 42 : étape 1 de l'addition de 16 nombres sur 4 processeurs MIMD en réseau

Etape 2: addition des sommes partielles sur les processeurs individuels (Figure 43).

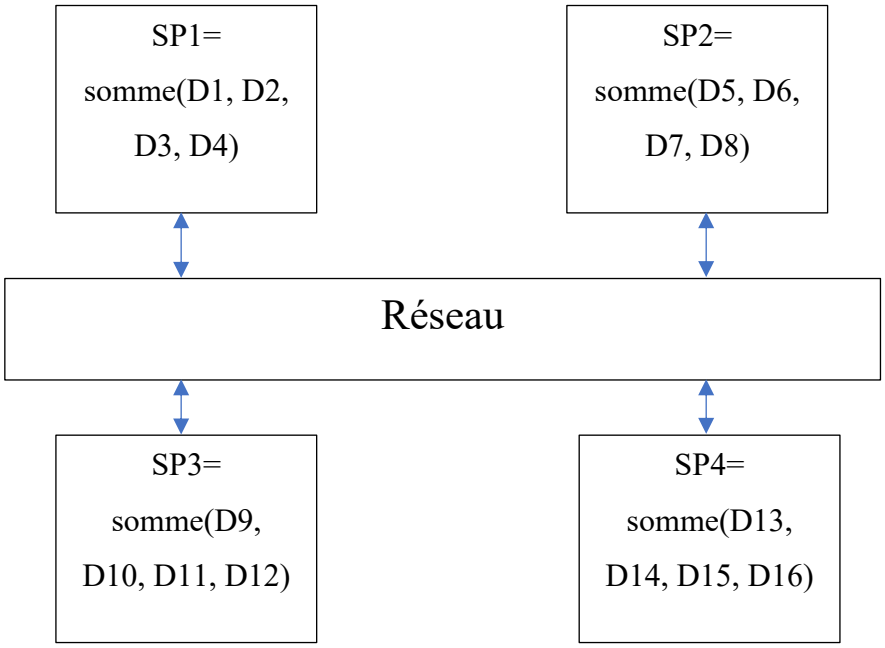


Figure 43 : étape 2 de l'addition de 16 nombres sur 4 processeurs MIMD en réseau

Etape 3: addition des sommes partielles, l'accès au réseau d'interconnexion pour communiquer les données. La moitié des processeurs envoie des données, l'autre moitié les reçoit et effectue les sommes partielles (Figure 44). En raison des temps d'exécution qui diffèrent, les récepteurs doivent être suspendus jusqu'à la réception des données.

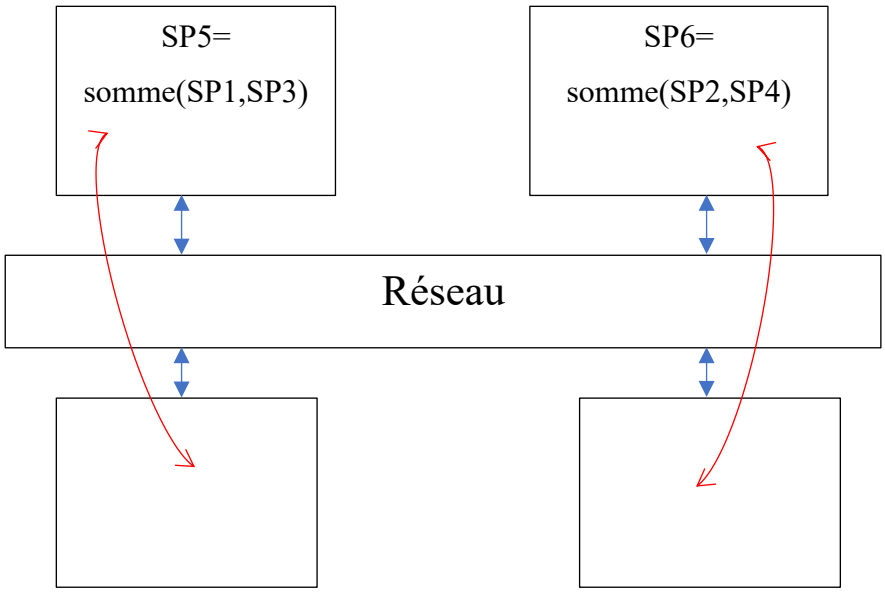


Figure 44 : étape 3 de l'addition de 16 nombres sur 4 processeurs MIMD en réseau

Etape 4: addition des sommes partielles à nouveau et le résultat final est obtenu (Figure 45).

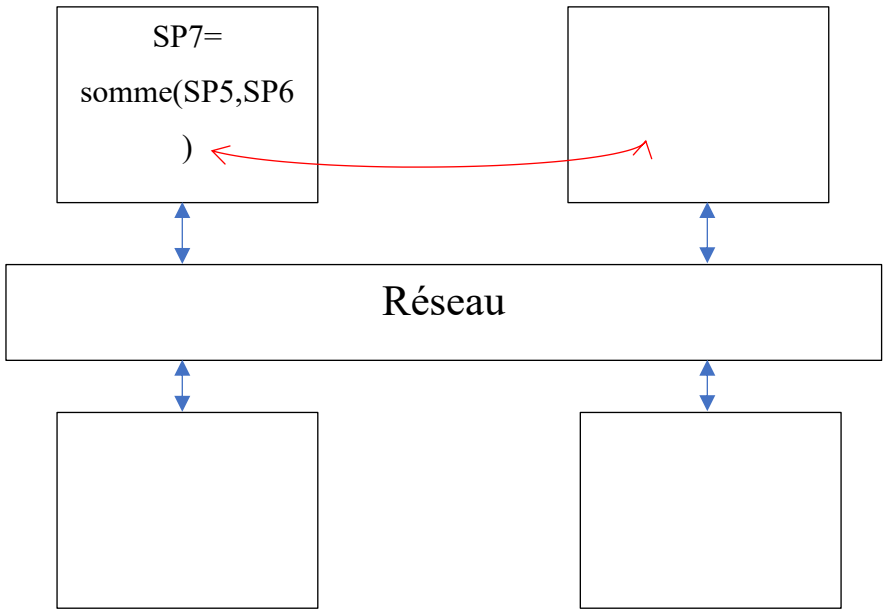


Figure 45 : étape 4 de l'addition de 16 nombres sur 4 processeurs MIMD en réseau.

Le programme du calcul de la somme dans ce cas est le suivant :

/ calcul des sommes initiales partielles*

Somme = 0

For (i = 0; i < 4 ; i = i + 1)

Somme = Somme + A1 [i];

/ diviser pour régner pour obtenir le résultat final*

Limite = 4

Moitie = 4 / 4 processeurs dans ce réseau MIMD*

Repeat

Moitie = Moitie / 2

if (Pn >= Moitie && Pn < Limite)

send (Pn - Moitie , Somme) / envoyer la somme à Pn/2 à travers le réseau*

if (Pn < Moitie)

Somme = Somme + Receive (); / suspendre jusqu'à ce que la donnée soit reçue*

Limite = Moitie

Until (moitié == 1);

Programme 4 : Programme d'addition MIMD réseau.

5. Questions de révision

Question 1 : Comparer les deux architectures SISD et SIMD.

Question 2 : Comparer les deux architecture Cluster Vs processeurs SMP.

Question 3 : Donner la définition d'un ordinateur parallèle.

Question 4 : Définir la Taxonomie de Flynn.

Consulter la section exercices pour plus de questions.

Solutions :

Question 1 : Comparaison SISD vs SIMD

- **Similarités:**

On peut dire que les deux architectures sont caractérisées par :

  - Une unité de contrôle unique.
  - Une seule copie du programme existe.
  - Une exécution synchrone du programme.
  - Le SIMD est conceptuellement identique au SISD.
  - Des unités fonctionnelles multiples s'exécutent.

- **Dissimilarité :**

La principale dissimilarité est que :

Dans une architecture SISD, le flux d'instruction opère sur le même flux de données alors que dans une architecture SIMD, le flux d'instruction opère sur des flux de données différents.

Question : Comparaison Cluster vs processeurs SMP

Similarité :

- Les deux architectures offrent des configurations avec plusieurs processeurs,
- Les deux présentent des modèles commercialisés,

Avantages du SMP:

- Plus facile à gérer et à configurer,
- Plus proche du modèle de processeur unique (programmes plus répandus);
- Occupe moins d'espace et consomme moins d'énergie.

Avantages du cluster :

- Les clusters sont supérieurs aux SMP en termes de scalabilité incrémentale et absolue, par exemple SMP Power Challenge de Silicon Graphics : 64 processeurs R10000 dans un seul système. Au-delà de ce nombre, les performances se dégradent de manière significative.
- Le cluster est supérieur en termes de disponibilité (possibilité de redondance).

Conclusion : les approches à base de clusters se répandent de plus en plus.

Question 3 : Les ordinateurs parallèles sont des machines qui comportent une architecture parallèle, constituée de plusieurs processeurs identiques, ou non, qui concourent au traitement d'une application.

Question 4 : La taxonomie de Flynn est une classification des architectures d'ordinateur, proposée par Michael Flynn en 1966. Elle est basée sur : (1) la nature du flux d'instructions exécuté par l'ordinateur et (2) la nature du flux de données sur lesquels opèrent ces instructions. Les quatre catégories résultantes de cette classification sont : SISD (Single Instruction stream Single Data stream), SIMD (Single Instruction stream Multiple Data stream), MISD (Multiple Instruction stream Single Data stream) et MIMD (Multiple Instruction stream Multiple Data stream).

6. Conclusion

Dans ce chapitre, le principe de fonctionnement des processeurs parallèles a été entamé en présentant les aspects architecturaux de chaque classe de processeurs, sa proche de programmation ainsi que des exemples illustratifs.

Série d'exercices

Exercice 1:

Le parallélisme consiste à mettre en œuvre des architectures et des algorithmes spécialisés permettant de traiter des informations de manière simultanée. Cela a pour but de réaliser le plus grand nombre d'opérations en un temps le plus petit possible. Aujourd'hui, il existe plusieurs types d'ordinateurs (ou de *processeurs*) parallèles, tels que :

1. Les processeurs SMP :
    a. Donnez une description détaillée des processeurs SMP.
    b. Les processeurs SMP appartiennent à quelle architecture selon la classification de Flynn ?
2. Les Clusters :
    a. Donnez une description détaillée des clusters.
    b. Les clusters appartiennent à quelle architecture selon la classification de Flynn ?

Exercice 2 :

On a besoin de concevoir un ordinateur spécialisé pour détecter automatiquement et à très grande vitesse l'immatriculation des voitures lors l'accès à la porte d'entrée d'un parking en utilisant une caméra.

1. Quelle est la meilleure architecture parallèle pour ce type d'opérations ? expliquez.

Supposant maintenant que ce parking possède plusieurs portes d'entrée, chacune est équipée par des caméras de contrôle d'accès.

2. Dans ce cas, quelle est la meilleure architecture pour ce type de traitement ? expliquez.

Exercice 3 :

1. Donnez un schéma détaillé des différentes organisations des processeurs selon la taxonomie de Flynn.
2. Citez quatre inconvénients du calcul parallèle.
3. Dans quelle classe(s) d'architecture parallèle existe(ent) une mémoire privée ? justifiez.
4. Une architecture pipelinée est-elle une architecture parallèle ? justifiez.

Exercice 4 :

Donner une brève description de :

(1) MPI, (2) OpenMP, (3) MPI_init(), (4) MPI_Finalize(), (5) MPI_Comm_rank(), (6) MPI_Comm_size(), (7) ≠Pragma omp parallel, (8) ≠Pragma omp critical .

Exercice 5

Le mode de communication des processus dans un environnement parallèle dépend fortement du type de l'architecture hardware du système.

1. Citer les différents modes de communication vus au cours.
2. Donner une brève description ainsi que les avantages et les inconvénients de chaque mode.

Exercice 6 :

MIMD est l'un des quatre modes de fonctionnement défini par la taxonomie de Flynn et désigne les machines multi-processeurs dont chaque processeur exécute son code de manière asynchrone et indépendante. Pour assurer la cohérence des données, il est souvent nécessaire de synchroniser les processeurs entre eux, les techniques de synchronisation dépendent de l'organisation de la mémoire.

1. Expliquer les différents modes d'organisations de la mémoire dans les architectures MIMD ?
2. Comparer ces organisations ?
3. Donner un exemple de processeur pour chaque organisation ?
4. Une dernière organisation a été proposée permettant d'améliorer les performances des architectures MIMD. Expliquer ?

Exercice7 :

La mémoire cache est une mémoire plus rapide et plus proche du processeur auquel elle sert des données et des instructions. Son rôle est d'économiser les échanges incessants entre le processeur et la mémoire vive.

1. Quel est le rôle du cache dans une architecture MIMD à mémoire partagée ?
2. Existe-il dans une architecture MIMD à mémoire distribué ? Expliquer?

Exercice 8 :

Le calcule parallèle permet d'accélérer efficacement les traitements. Cependant, cette technologie a généré plusieurs problèmes telle que l'incohérence de données.

1. Définir l'incohérence de données ?
2. Dans quel classe(s) des architectures parallèles persiste ce problème ?
3. C'est quoi la solution ? Cette solution est prise en charge par le système ? expliquer ?

Série des Travaux pratiques

Les exercices présentés dans cette section concernent la programmation parallèle avec MPI [18].

Exercices avec corrigés

Exercice 1 :

Ecrire une version du programme MPI « Hello Word » qui, pour chaque processus, affiche son numéro et le nombre total de processus.

Exercice 2 :

Modifier le programme précédent pour que les processus pairs n'affichent pas le même message que les processus impairs.

Exercice 3 :

Ecrire un programme MPI dans lequel le processus 1 envoie un tableau de 10 entiers au processus 4.

Exercice 4 :

Ecrire un programme dans lequel un jeton tourne dans un anneau de processeur. Initialement le jeton est dans le processeur 0, puis il passe d'un processeur à son successeur jusqu'au retourner au processeur 0.

Exercices sans corrigés

Exercice 5 :

Ecrire un programme MPI dans le quel le processeur 0 envoie la 10^{eme} valeur générée aléatoirement au processeur 1.

Exercice 6 :

Ecrire un programme qui calcule la somme des nombres premiers inférieurs à une valeur donnée n (99999). Dans un premier temps, aucune directive MPI ne sera utilisée, le programme étant alors exécuté séquentiellement.

• Calculer le temps d'exécution.

Exercice 7 :

Paralléliser l'exercice 6 à l'aide d'une boucle parallèle MPI.

• Calculer le temps d'exécution et comparer avec celui du programme séquentiel.

Solutions des travaux pratiques

Exercice 1 :

```
#include <mpi.h>
void main(argv, argc)
int argv;
char *argc[];
{
        int taille, numéro;
        MPI_Init(&argv, &argc);
        MPI_Comm_rank(MPI_COMM_WORLD, &numéro);
        MPI_Comm_size(MPI_COMM_WORLD, &taille);
        printf("Hello world du processus  %d parmi %d\n", numéro, taille);
        MPI_Finalize();
}
```

Exercice 2 :

```
#include <mpi.h>
void main(argv, argc)
int argv;
char *argc[];
{
```

```
        int taille, numéro;
        MPI_Init(&argv, &argc);
        MPI_Comm_rank(MPI_COMM_WORLD, &numéro);
        MPI_Comm_size(MPI_COMM_WORLD, &taille);
        printf("Hello world du processus %d parmi %d (", numéro, taille);
        if (num % 2)
                printf("pair)\n");
        else
                printf("impair )\n");
                MPI_Finalize();
}
```

Exercice 3

```
#include <mpi.h>
void main(argv, argc)
int argv;
char *argc[];
{
        int taille, numéro;
        int tab[10], i;
        MPI_Status statut;
        MPI_Init(&argv, &argc);
        MPI_Comm_rank(MPI_COMM_WORLD, &numéro);
        MPI_Comm_size(MPI_COMM_WORLD, &taille);
```

```
        printf("Hello world from %d out of %d\n", numéro, taille);
        if (numéro == 0) {
                for (i=0; i<10; i++)
                tab[i] = i;
                MPI_Send(tab, 10, MPI_INT, 1, 1000, MPI_COMM_WORLD);
                printf("0 a envoye 10 valeurs a 1\n");
        }
        else {
                MPI_Recv(tab, 10, MPI_INT, 0, 1000, MPI_COMM_WORLD,
                &statut);
                printf("1 a recu 10 valeurs de 0\n");
                for (i=0; i<10; i++)
                printf("pour 1 tab[%d]=%d\n", i, tab[i]);
        }
        MPI_Finalize();
}
```

Exercice 4 :

```
#include <mpi.h>
void main(argv, argc)
int argv;
char *argc[];
{
        int taille, numéro;
        int jeton, gauche, droit;
        MPI_Status statut;
```

```
        MPI_Init(&argv, &argc);
        MPI_Comm_rank(MPI_COMM_WORLD, &numéro);
        MPI_Comm_size(MPI_COMM_WORLD, &taille);
        gauche = (numéro - 1 + taille)%taille;
        droit = (numéro + 1)%taille;
        printf("Hello world from %d out of %d (%d,%d))\n", numéro, taille,
        gauche, droit);
        if (numéro == 0) {
                jeton = 12;
                MPI_Send(&jeton, 1, MPI_INT, droit, 1,
                MPI_COMM_WORLD);
                printf("%d a envoye %d a %d\n", numéro, jeton, droit);
                MPI_Recv(&jeton, 1, MPI_INT, gauche, 1,
                MPI_COMM_WORLD, &statut);
                printf("%d a recu %d de %d\n", numéro, jeton, gauche);
        }
        else {
        MPI_Recv(&jeton, 1, MPI_INT, gauche, 1, MPI_COMM_WORLD,
        &statut);
        printf("%d a recu %d de %d\n", numéro, jeton, gauche);
        jeton++;
        MPI_Send(&jeton, 1, MPI_INT, droit, 1, MPI_COMM_WORLD);
        printf("%d a envoye %d a %d\n", numéro, jeton, droit);
        }
}
```

Conclusion

Le domaine du traitement parallèle concerne les méthodes architecturales et algorithmiques permettant d'améliorer les performances des ordinateurs numériques ou d'autres critères tels que la rentabilité et la fiabilité grâce à diverses formes de concurrence. Même si le calcul concurrent existe depuis les premiers jours des ordinateurs numériques, ce n'est que récemment qu'il a été appliqué d'une manière permettant d'obtenir de meilleures performances, ou un meilleur rapport coût-efficacité, par rapport aux superordinateurs vectoriels.

Ce polycopié représente d'une manière simplifiée les principes de bases liées aux architectures parallèles en abordant brièvement leurs concepts, leurs caractéristiques, la classification de ce type d'architectures ainsi que l'approche de programmation de chacune des classes.

Références Bibliographique

1. M. Abd-El-Barr, H. El-Rewini. Fundamentals of Computer Organization and Architecture. John Wiley & Sons, Inc. 2005.
2. H. Bersini, M. P. Spinette, R. Spinette. Les Fondements de l'informatique, Vuibert, 2008.
3. Philippe Breton, Une histoire de l'informatique. Paris, éditions du Seuil, coll. Points Sciences. 1990.
4. Djamal Rebaine, Historique et évolution des ordinateurs, Université du Québec.
5. Philippe Dubois, l'histoire de l'informatique, MO5.COM, 2021.
6. TERMIUM Plus®, la banque de données terminologiques et linguistiques du gouvernement du Canada, 2021.
7. M. M. Pierre. 2. Le programme Ordinateurs de cinquième génération : contenu, réception et analyses. In:Ebisu, n°5, 1994. pp. 45-87.
8. Jean-Pierre Briot, Rapport CNRS-JAPON : 5th Generation Project. 1992.
9. M. Koudil, Architectures parallèles, université de bejaia. 2005.
10. V. Rajaraman, C. S. R. Murthy. Parallel Computers Architecture and Programming, 2016.
11. D. Culler, J.P. Singh, A. Gupta. Parallel Computer Architecture A Hardware Software Approach. 1998.
12. Behrooz Parhami - Introduction to parallel processing: algorithms and architectures, Springer.1999.
13. A. D. George - Microprocessor-based Parallel Architecture for Reliable Digital Signal Processing Systems, CRC Pr I LLc. 2017.
14. B. T. Smith. parallel strategies, University of New M. 1998.
15. K. Hwang, N. Jotwani. Advanced Computer Architecture Parallelism, Scalability, Programmability. 1993.
16. H. El-Rewini, M. Abd-El-Barr. Advanced computer architecture and parallel processing. 2005.
17. R. Trobec, B. Slivnik, P. Bulić, B. Robič. Introduction to Parallel Computing: From Algorithms to Programming on State-of-the-Art Platforms. 2018.
18. M. Nakechbandi. Modèles discrets : évolution, auto-organisation. Université Du Havre. 2008/2009.

Sites web :

w1. https://spectrum.ieee.org/alan-turing-how-his-universal-machine-became-a-musical-instrument

w2. https://fr.wikipedia.org/wiki/Electronic_Discrete_Variable_Automatic_Computer

w3. https://fr.wiktionary.org/wiki/carte_perfor%C3%A9e

w4. https://fr.wikipedia.org/wiki/Tube_%C3%A9lectronique

w5. https://electrosttissemsilt.files.wordpress.com/2017/04/technologie-et-fabrication-des-circuits-intc3a9grc3a9s-partie-01.pdf

w6. https://www.wikiwand.com/fr/IBM_360_et_370

w7. https://www.cite-telecoms.com/blog/histoire/200-ans-de-telecoms/lere-des-convergences/la-puce-le-micro-ordinateur/