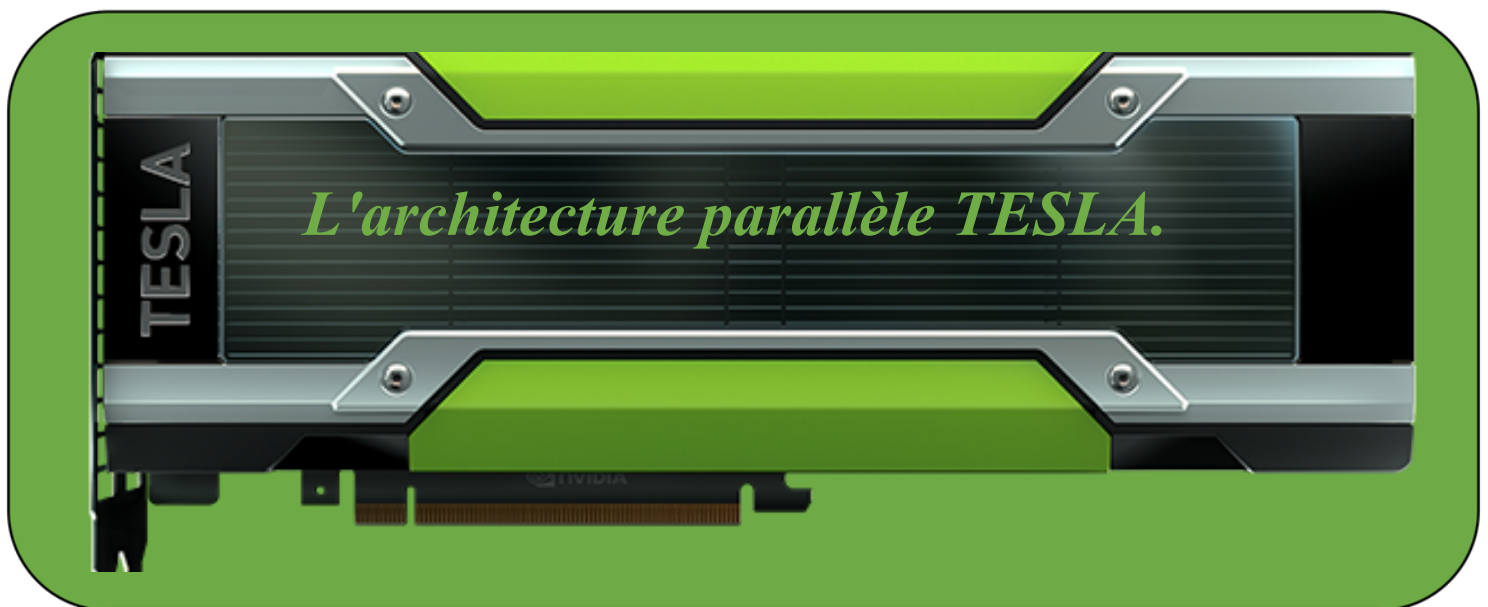


*République Algérienne Démocratique et Populaire*  
*Ministère de l'Enseignement Supérieur et de la Recherche Scientifique*



*Université des Sciences et de la Technologie*  
*- Houari Boumediene –*  
*Faculté d'Informatique*

*Département Des Systèmes Informatique*



*Module : Programmation multi-cœurs.*

*Niveau : Master 2 HPC.*

*Préparer par:*

*Enseignant:*

*BOULAHIA Yasmine.*

*Mr.H.SAADI.*

*ZERKOUK K haoula.*

*SAIDANI Cerine.*

*Année universitaire : 2025/2026*

## Table des matières :

Résumé.....	5
Introduction générale.....	6
1. Contexte scientifique et technologique.....	6
2. Problématique et questions de recherche.....	6
3. Objectifs du mémoire .....	7
4. Structure du mémoire .....	7
Chapitre 1 : Principes fondamentaux du calcul parallèle.....	9
Introduction .....	10
Historique du calcul haute performance .....	10
Parallélisme de données et parallélisme de tâches .....	12
Conclusion .....	12
Chapitre 2 : Évolution des GPU avant Tesla.....	13
Introduction .....	14
Pipeline graphique traditionnel et GPU fixes .....	14
Conclusion .....	17
Chapitre 3 : Panorama du calcul GPU en 2007.....	18
Introduction .....	19
Les approches GPGPU avant CUDA .....	19
Avantages de ces approches .....	19
Limites majeures.....	19
Les architectures GPU disponibles en 2007 .....	20
Conclusion .....	25
Chapitre 4 : Positionnement scientifique et industriel de Tesla. ....	26
Introduction .....	27
Contexte scientifique avant Tesla : le besoin du changement. ....	27
Positionnement technologique : Tesla comme rupture architecturale.....	28
Introduction d'un modèle mémoire adapté au HPC .....	30
Positionnement logiciel : Tesla + CUDA = révolution programmatrice.....	31
Positionnement industriel : l'arrivée des GPU dans les datacenters .....	32
Début du GPU dans les supercalculateurs TOP500 .....	34
Conclusion .....	34

CHAPITRE 5 : Présentation générale de l'architecture Tesla. ....	35
Introduction .....	36
Contexte de développement de la génération G80 .....	36
Objectifs technologiques de Tesla .....	36
Caractéristiques globales de l'architecture Tesla .....	36
Innovations majeures introduites .....	37
Conclusion .....	37
CHAPITRE 6 : Architecture matérielle interne de Tesla. ....	38
Introduction .....	39
Organisation globale du GPU G80 .....	39
Conclusion .....	41
CHAPITRE 7 : Hiérarchie mémoire de Tesla. ....	42
Introduction .....	43
Mémoire globale .....	43
Mémoire partagée (Shared Memory) .....	43
Registres .....	44
Mémoire constante et texture .....	45
Bande passante, cohérence et limitations .....	45
Conclusion .....	46
CHAPITRE 8 : Limites techniques de Tesla. ....	47
Introduction .....	48
Divergence des warps .....	48
Limitations mémoire .....	48
Limitations du modèle CUDA 1.0 .....	48
Consommation énergétique et thermique .....	49
Conclusion .....	49
CHAPITRE 9 : Applications majeures basées sur l'architecture Tesla. ....	50
Introduction .....	51
Simulation numérique et physique computationnelle .....	51
Calcul scientifique et HPC .....	51
Traitement d'images et vision par ordinateur .....	52
Apprentissage automatique (ML) et premiers usages en IA .....	52
Informatique graphique et rendu avancé .....	53

Conclusion .....	53
CHAPITRE 10 : Impact et héritage de l'architecture Tesla.....	54
Introduction .....	55
Contribution à l'évolution du GPGPU .....	55
Influence sur les générations ultérieures de GPU .....	55
Influence sur les supercalculateurs .....	55
Impact sur l'intelligence artificielle moderne .....	56
Impact industriel durable .....	56
Conclusion .....	56
Conclusion générale .....	57
Bibliographie et webographie. ....	58

## Table des figures:

1 Illustration pratique de la classification de Flynn (CPU – SIMD – GPU).	11
2 Illustration pratique de la classification de Flynn (CPU – SIMD – GPU).	11
3 Data parallelism vs Task parallelism.	12
4 GeForce 256 la plus ancienne carte graphique Nvidia 1999.	14
5 Schéma fonctionnel du Vertex Shader Core.	15
6 Schéma fonctionnel du pixel shaders Core.	15
7 Carte 3D avec pixels et vertex shaders non-unifiés.	20
8 Architecture de la GeForce 6800.	21
9 Carte 3D avec pixels et vertex shaders unifiés.	22
10 GPU vs CPU.	23
11 Besoin d'une architecture HPC dédiée.	24
12 L'architecture SIMT[7].	28
13 Comment sont exécutées les threads en CUDA.	29
14 La taille des GPU exige la scalabilité de CUDA.	30
15 L'exécution d'un programme CUDA.	31
16 NVIDIA Tesla C1060.	32
17 NVIDIA Tesla S1070.	32
18 Performances des cartes Tesla P100 et V100.	34
19 GPU G80.	36
20 Chaîne de traitement pour le calcul parallèle des threads sur le G80.	39
21 L'architecture parallèle du G80 : MIMD 8 voies × SIMD 16.	40
22 L'architecture unifiée GPU Tesla pour le graphisme et le calcul.	43
23 La communication hôte - «device» [8].	44
24 Multiprocesseur de flux (SM) dans le GPU Tesla.	45
25 La hiérarchie de mémoire et de threads [8].	46
26 Comparaison des hiérarchies mémoire et des capacités des GPU Tesla, Fermi, Kepler et Maxwell. [12]	49
27 Simulation Boston Scientific : distribution du champ électrique à 64 MHz. [13]	51
28 Imagerie médicale : tomosynthèse numérique. [13]	52
29 Simulation de circuits neuronaux : machines évoluées. [13]	53

## Résumé

Ce mémoire présente une étude détaillée de l'architecture **NVIDIA Tesla (G80)**, le processeur graphique qui a marqué la **transition** entre les cartes 3D et le **calcul haute performance (HPC)** dédié<sup>1</sup>. Il analyse son contexte d'apparition et son impact en tant que **première architecture unifiée** et entièrement programmable via le modèle **CUDA**.

L'objectif principal est de comprendre comment l'approche **SIMT (Single Instruction, Multiple Threads)** de Tesla exploite un **parallélisme massif** pour accélérer des applications scientifiques, industrielles et d'intelligence artificielle. Le mémoire détaille l'organisation en **Streaming Multiprocessors (SM)**, la hiérarchie mémoire, et le rôle du **warp** dans la gestion de l'exécution parallèle.

La méthodologie repose sur une analyse comparative avec les architectures CPU traditionnelles et une étude des performances. Les résultats confirment l'accélération significative pour les applications fortement parallèles, tout en identifiant les **faiblesses structurelles** de cette première génération, notamment une faible efficacité en **double précision (FP64)** et l'absence de **cache L1/L2** sophistiqué.

En conclusion, ce travail démontre que Tesla constitue une **étape fondatrice** pour le HPC, ayant établi le modèle des GPU modernes (Fermi, Kepler, etc.) et redéfini l'avenir du calcul intensif.

# Introduction générale

L'évolution rapide des technologies de calcul a profondément transformé les domaines scientifiques, industriels et numériques. Au début des années 2000, la croissance des performances des processeurs centraux (CPU) a progressivement ralenti en raison des limites physiques liées à la fréquence d'horloge et à la consommation énergétique.

Cette stagnation a conduit à l'émergence de nouveaux paradigmes de calcul reposant sur le parallélisme massif.

Dans ce contexte, les processeurs graphiques (GPU), initialement conçus pour le rendu graphique, se sont imposés comme une solution prometteuse pour l'accélération des applications générales.

L'architecture **NVIDIA Tesla**, introduite en 2007, marque ainsi une étape déterminante dans cette transition en faisant du GPU un processeur pleinement programmable dédié au calcul scientifique.

## 1. Contexte scientifique et technologique

Avant l'apparition de Tesla, les architectures parallèles étaient principalement dominées par les systèmes **SMP (Symmetric Multiprocessing)** et **MPP (Massively Parallel Processors)**. Parallèlement, les premiers GPU programmables offraient une certaine flexibilité, mais restaient limités par des modèles fondés sur les shaders et des langages peu adaptés au calcul général.

L'arrivée de Tesla s'inscrit donc dans une dynamique scientifique et technologique visant à exploiter le parallélisme massif pour surmonter les limites des architectures CPU traditionnelles. Grâce à l'introduction du modèle de programmation **CUDA**, Tesla a ouvert la voie à une nouvelle génération de solutions HPC.

## 2. Problématique et questions de recherche

L'apparition de l'architecture Tesla soulève plusieurs questions fondamentales que ce mémoire vise à explorer :

- Comment l'architecture Tesla a-t-elle été conçue pour exploiter efficacement le parallélisme massif ?
- En quoi se distingue-t-elle des architectures CPU, SMP et MPP contemporaines ?
- Quels sont les mécanismes architecturaux et mémoire permettant d'atteindre des performances élevées ?
- Quelles sont les limites inhérentes à Tesla et comment ont-elles influencé les architectures GPU ultérieures ?
- Quel impact Tesla a-t-elle eu sur l'évolution du calcul scientifique, de l'intelligence artificielle et du HPC ?

### 3. Objectifs du mémoire

Ce mémoire poursuit les objectifs suivants :

Présenter et analyser en profondeur l'architecture NVIDIA Tesla (2007) en mettant en évidence ses caractéristiques structurelles et ses innovations<sup>24</sup>.

Étudier son modèle de programmation CUDA, ainsi que les principes d'exécution des threads, warps et blocs<sup>25</sup>.

Évaluer ses performances à travers des travaux expérimentaux ou une analyse comparative avec d'autres architectures parallèles<sup>26</sup>.

Identifier ses limites et contraintes, notamment en matière de mémoire, de synchronisation et de gestion des ressources<sup>27</sup>.

### 4. Structure du mémoire

Ce mémoire s'articule en **cinq parties principales**, chacune regroupant plusieurs chapitres :

**Partie I — Fondements théoriques et cadre conceptuel** Cette partie présente les principes du calcul parallèle et l'évolution des architectures GPU avant l'arrivée de Tesla.

- Chapitre 1 : Principes fondamentaux du calcul parallèle
- Chapitre 2 : Évolution des GPU avant Tesla

**Partie II — État de l'art du GPGPU et positionnement de Tesla** Elle analyse les approches existantes du calcul GPU avant 2007 et situe Tesla dans son contexte scientifique et industriel.

- Chapitre 3 : Panorama du calcul GPU en 2007
- Chapitre 4 : Positionnement scientifique et industriel de Tesla

**Partie III — L'architecture NVIDIA Tesla (2007)** Cette partie détaille la conception interne de Tesla, son modèle d'exécution CUDA, sa hiérarchie mémoire et son organisation matérielle.

- Chapitre 5 : Présentation générale de Tesla
- Chapitre 6 : Architecture matérielle interne
- Chapitre 7 : Hiérarchie mémoire
- Chapitre 8 : Performances théoriques et pratiques
- Chapitre 9 : Limites techniques de Tesla



**Partie IV — Applications et impact scientifique** Elle présente les domaines d’application de Tesla ainsi que son héritage et son influence sur les architectures GPU moderne.

- Chapitre 10 : Applications majeures basées sur Tesla
- Chapitre 11 : Impact et héritage de l’architecture Tesla

**Partie V — Discussion et conclusion générale** Cette partie propose une analyse critique de Tesla, revient sur la problématique initiale et suggère des perspectives pour les recherches futures.

# Chapitre 1 : Principes fondamentaux du calcul parallèle.

## Introduction

Ce chapitre a pour objectif de présenter les fondements théoriques du calcul parallèle, son évolution historique, les principales classifications d'architectures et les types de parallélisme. Il pose ainsi le cadre conceptuel nécessaire pour comprendre l'architecture NVIDIA Tesla, étudiée dans les chapitres suivants.

## Historique du calcul haute performance

Le calcul haute performance (HPC) a émergé dès le milieu du XX<sup>e</sup> siècle, motivé par la nécessité de résoudre des problèmes scientifiques et techniques complexes tels que la simulation nucléaire, la météorologie ou la modélisation moléculaire.

Les premiers supercalculateurs étaient des machines vectorielles, capables de traiter de longues séquences de données simultanément.

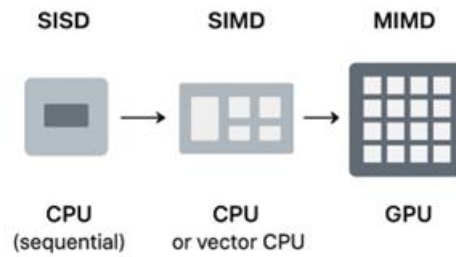
Dans les années 1970 et 1980, l'émergence du parallélisme massif a conduit à la création des architectures SMP (Symmetric Multiprocessing) et MPP (Massively Parallel Processors), permettant d'utiliser plusieurs processeurs pour exécuter des tâches simultanément.

Avec le début des années 2000, les GPU programmables ont ouvert un nouveau paradigme : l'exploitation du parallélisme massif pour des applications générales (GPGPU), donnant naissance à des architectures telles que NVIDIA Tesla.

### • Classifications des architectures parallèles

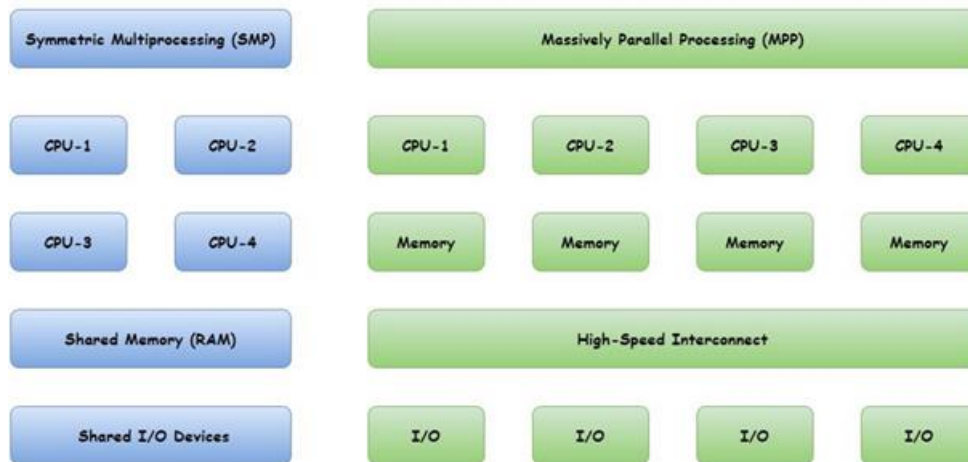
Selon la classification de Flynn (1966), les architectures parallèles se distinguent par le nombre de flux d'instructions et de données :

- **SISD (Single Instruction, Single Data)** : un flux d'instructions traite une seule donnée à la fois, modèle classique des CPU séquentiels.
- **SIMD (Single Instruction, Multiple Data)** : un flux d'instructions contrôle plusieurs unités de traitement simultanément, chacune opérant sur des données différentes, typique des processeurs vectoriels et GPU.
- **MISD (Multiple Instruction, Single Data)** : plusieurs flux d'instructions exécutent des opérations sur une même donnée, modèle rare en pratique.
- **MIMD (Multiple Instruction, Multiple Data)** : plusieurs flux d'instructions traitent simultanément différentes données, utilisé dans les supercalculateurs et clusters modernes.



### 1 Illustration pratique de la classification de Flynn (CPU – SIMD – GPU).

- **Modèles SMP, MPP et architectures massivement parallèles**
- **SMP** (Symmetric Multiprocessing) : plusieurs processeurs partagent la même mémoire et le même système d'E/S. Chaque processeur est équivalent et exécute des tâches parallèles de manière coopérative.
- **MPP** (Massively Parallel Processors) : de nombreux processeurs indépendants avec mémoire locale, interconnectés par un réseau rapide, exécutant chacun des tâches spécifiques.



### 2 Illustration pratique de la classification de Flynn (CPU – SIMD – GPU).

- Architectures **massivement parallèles** : regroupent des milliers de cœurs ou processeurs capables d'exécuter simultanément de nombreuses tâches fines.

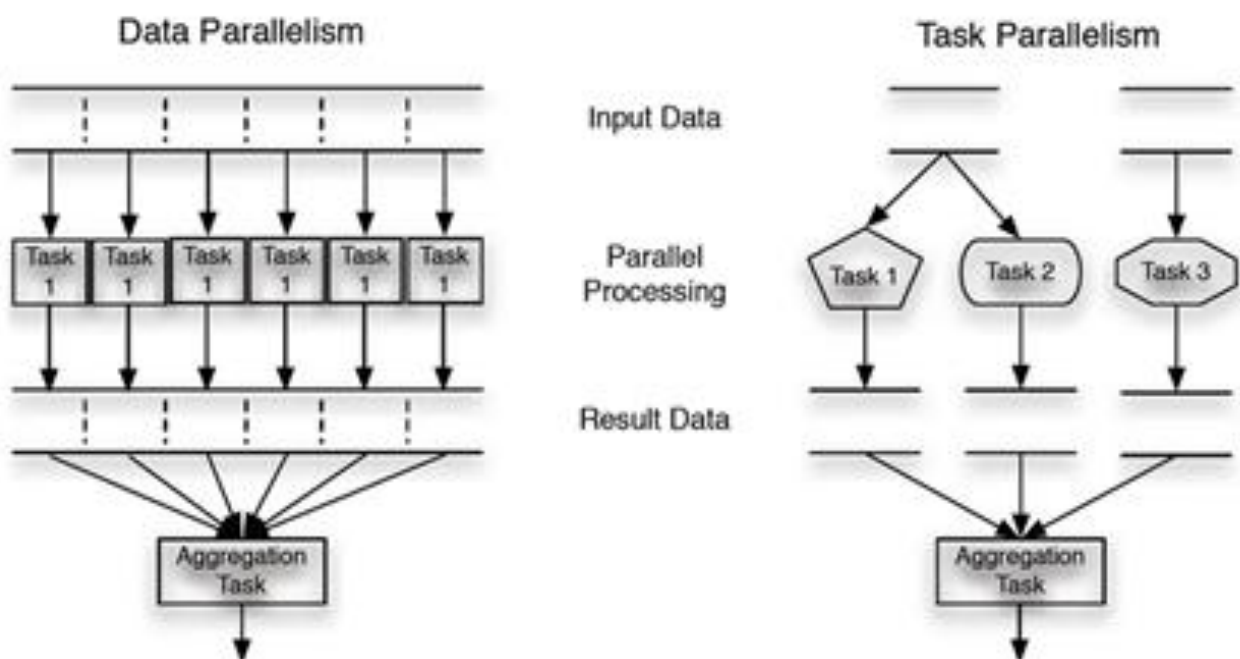
Les GPU modernes comme Tesla sont un exemple typique, combinant un grand nombre de multiprocesseurs de flux (SM) pour exploiter efficacement le parallélisme de données.

## Parallélisme de données et parallélisme de tâches

Parallélisme de données (**Data Parallelism**) : mêmes opérations appliquées simultanément sur des éléments différents de données (matrices, vecteurs, tableaux). C'est le parallélisme prédominant sur GPU.

Parallélisme de tâches (**Task Parallelism**) : différentes tâches exécutées simultanément sur des données distinctes<sup>63</sup>. Courant sur les clusters ou les CPU multi-threads<sup>64</sup>.

L'efficacité du calcul parallèle dépend de la capacité de l'architecture à réduire les dépendances entre tâches, minimiser la latence mémoire et optimiser la communication entre processeurs, des principes essentiels pour comprendre les choix architecturaux de Tesla<sup>65</sup>.



### 3 Data parallelism vs Task parallelism.

## Conclusion

Le calcul parallèle est la pierre angulaire des architectures haute performance modernes. L'évolution historique montre le passage des systèmes séquentiels aux architectures massivement parallèles, préparant le terrain pour l'avènement des GPU programmables. Les classifications des architectures et les types de parallélisme introduisent les concepts clés qui seront appliqués et approfondis dans l'étude de l'architecture NVIDIA Tesla.

## Chapitre 2 : Évolution des GPU avant Tesla.

## Introduction

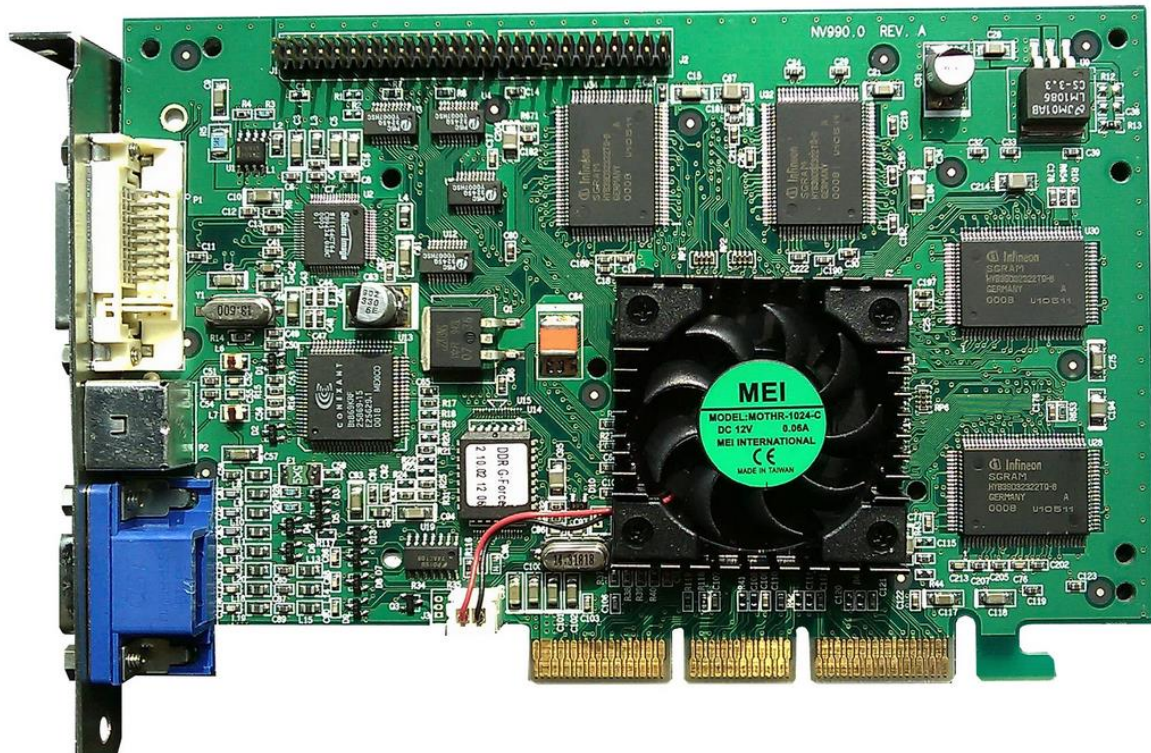
Avant l'architecture NVIDIA Tesla, les GPU ont évolué du simple rendu graphique vers des processeurs programmables capables d'effectuer des calculs généraux. Ce chapitre présente cette évolution, les principales limitations des GPU pré-Tesla et les besoins émergents qui ont conduit à la conception de Tesla.

## Pipeline graphique traditionnel et GPU fixes

Les premières cartes graphiques étaient conçues pour **accélérer le rendu 3D en temps réel**, avec un pipeline fixe :

- Transformation et projection des vertices : calcul des positions 3D des objets dans la scène.
- **Rasterisation** : conversion des primitives (polygones) en pixels.
- **Texturing et shading** : application des textures et calcul des couleurs.
- Frame buffer et sortie vidéo : stockage final des pixels à afficher.

Ces GPU fixes (ex. NVIDIA RIVA, GeForce 256 la première carte graphique NVIDIA publiée en 1999) étaient très efficaces pour le rendu, mais entièrement spécialisés : les développeurs ne pouvaient pas les utiliser pour effectuer des calculs scientifiques ou parallèles autres que ceux liés à la 3D. [R4]

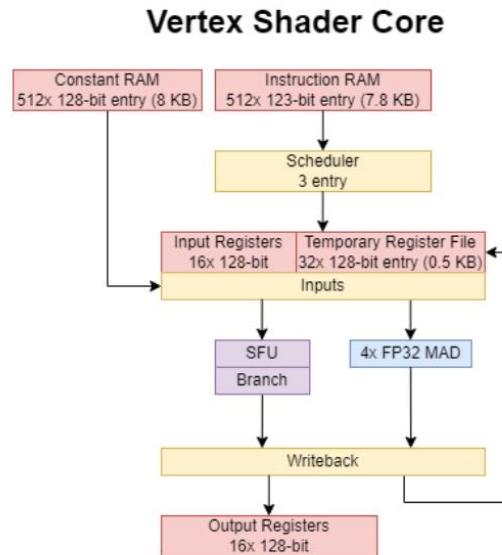


4 GeForce 256 la plus ancienne carte graphique Nvidia 1999.

- **GPU programmables (2001–2006)**

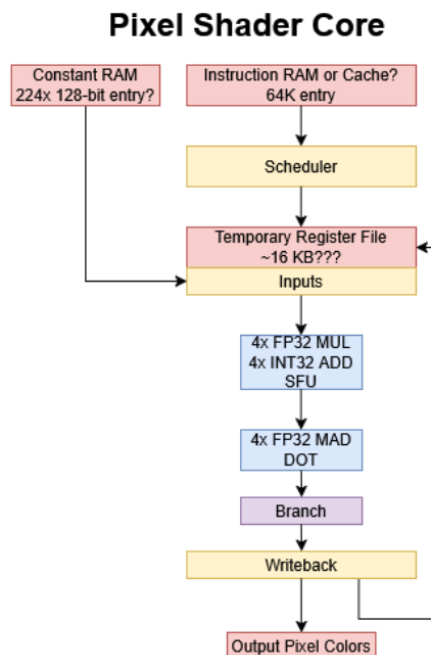
Avec l'arrivée des GPU programmables, comme les générations **GeForce FX et 6xxx**, NVIDIA a introduit des **vertex shaders et pixel shaders programmables** :

- Les **vertex shaders** permettaient de transformer les sommets selon des programmes définis par l'utilisateur.



##### 5 Schéma fonctionnel du Vertex Shader Core.

- Les **pixel shaders** autorisaient des calculs sur chaque pixel pour simuler des effets graphiques complexes. [5]



##### 6Schéma fonctionnel du pixel shaders Core.



Cette programmabilité a ouvert la voie au GPGPU (General-Purpose GPU Computing), mais le modèle restait limité :

- La **mémoire était segmentée et peu flexible** : accès global et local peu contrôlé.
- La **programmation nécessitait de détourner le pipeline graphique** pour des calculs non graphiques.
- La **gestion des threads et des parallélismes fins** était difficile, limitant les performances sur des tâches massivement parallèles.
- **Limites et contraintes des GPU pré-Tesla**

Malgré leur programmabilité, les GPU avant Tesla avaient plusieurs limites :

- **Complexité du développement** : il fallait coder en shaders graphiques, pas en langage de calcul général.
- **Hierarchie mémoire rigide** : pas de mémoire partagée pour les threads, latences importantes sur la mémoire globale.
- **Synchronisation limitée** : difficile de coordonner les calculs entre groupes de threads.
- Performance non optimale pour le HPC : divergence de flux et dépendances mémoire entraînaient des inefficacités.

Ces limites ont montré que, pour le HPC, un GPU devait être un processeur parallèle complet, programmable avec un modèle adapté aux milliers de threads simultanés.

- **Besoins émergents pour le HPC**

Plusieurs facteurs ont rendu indispensable la création d'un GPU adapté au calcul scientifique :

- **Stagnation des performances CPU** : augmentation limitée des fréquences et consommation énergétique croissante.
- **Applications HPC de plus en plus massives** : simulations physiques, modélisation numérique, analyse de données scientifiques.
- Nécessité d'une programmation simplifiée et d'un accès efficace à la mémoire pour exploiter le parallélisme massif.

Ces besoins ont directement conduit NVIDIA à développer l'architecture Tesla, qui combine programmabilité complète, hiérarchie mémoire optimisée et parallélisme massif.

## Conclusion

L'évolution des GPU avant Tesla illustre la **transition progressive du rendu graphique vers le calcul scientifique**. Les GPU programmables ont ouvert de nouvelles possibilités mais restaient limités par leur architecture et leur modèle de programmation. L'analyse des limites et besoins du HPC a permis de définir les exigences auxquelles l'architecture Tesla répondra, notamment en termes de parallélisme massif, hiérarchie mémoire flexible et programmation simplifiée via CUDA.

## Chapitre 3 : Panorama du calcul GPU en 2007.

## Introduction

En 2007, le domaine du calcul de haute performance connaît une transition majeure : les GPU, initialement conçus pour le rendu graphique, s'imposent progressivement comme des accélérateurs capables d'exécuter des calculs scientifiques massivement parallèles. Cette année représente un point charnière : la recherche explore activement le potentiel du GPGPU, tandis que l'industrie développe de nouvelles architectures pour dépasser les limites des pipelines graphiques programmables.

Ce chapitre présente un panorama des technologies, méthodes et architectures de calcul GPU disponibles jusqu'en 2007. Il analyse les premières approches GPGPU, leurs limites, et met en lumière l'environnement technologique dans lequel l'architecture NVIDIA Tesla apparaîtra.

## Les approches GPGPU avant CUDA

Avant 2007, programmer un GPU pour autre chose que du rendu graphique nécessitait d'utiliser le pipeline 3D. On parle de **GPGPU « détourné »**.

- **Utilisation des shaders comme unités de calcul**

Les GPU programmables permettaient d'écrire des **vertex shaders** et **pixel shaders**. Les chercheurs utilisaient ces unités pour effectuer des calculs généraux :

- Les données étaient représentées comme des textures.
- Les opérations se faisaient via des programmes de shading.
- Les résultats étaient stockés dans des render targets.

- **Les premières bibliothèques GPGPU sont :**

- BrookGPU (Stanford, 2003)
- Sh (University of Waterloo)
- CG de NVIDIA (limitée au shading)

## Avantages de ces approches

- Exploitation du parallélisme massif des pipelines graphiques.
- Accélération notable pour des calculs de type SIMD.
- Coût faible par rapport aux clusters CPU.

## Limites majeures

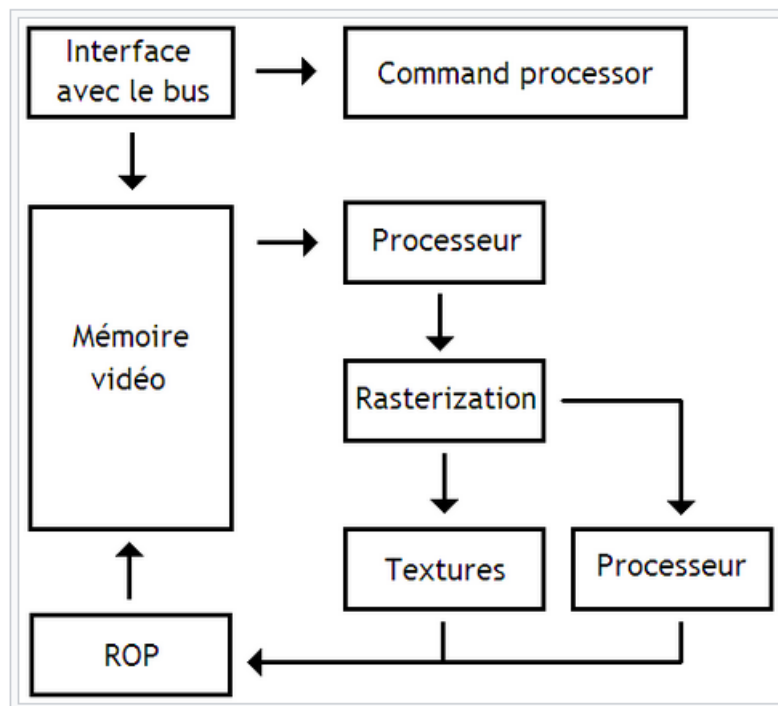
- Non-généralité : obligation d'exprimer l'algorithme comme un problème de rendu.
- Programmation très complexe : shaders, textures, buffers, pipeline rigide.

- Gestion mémoire limitée, difficile de gérer les structures de données complexes.
- Pas de débogage standard ni d'outils HPC.
- Synchronisation faible : impossible d'échanger efficacement entre threads.

Ces limites ont freiné l'adoption du GPGPU dans les domaines scientifiques.

## Les architectures GPU disponibles en 2007

En 2007, la majorité des GPU reposent sur des architectures à **shaders séparés**:



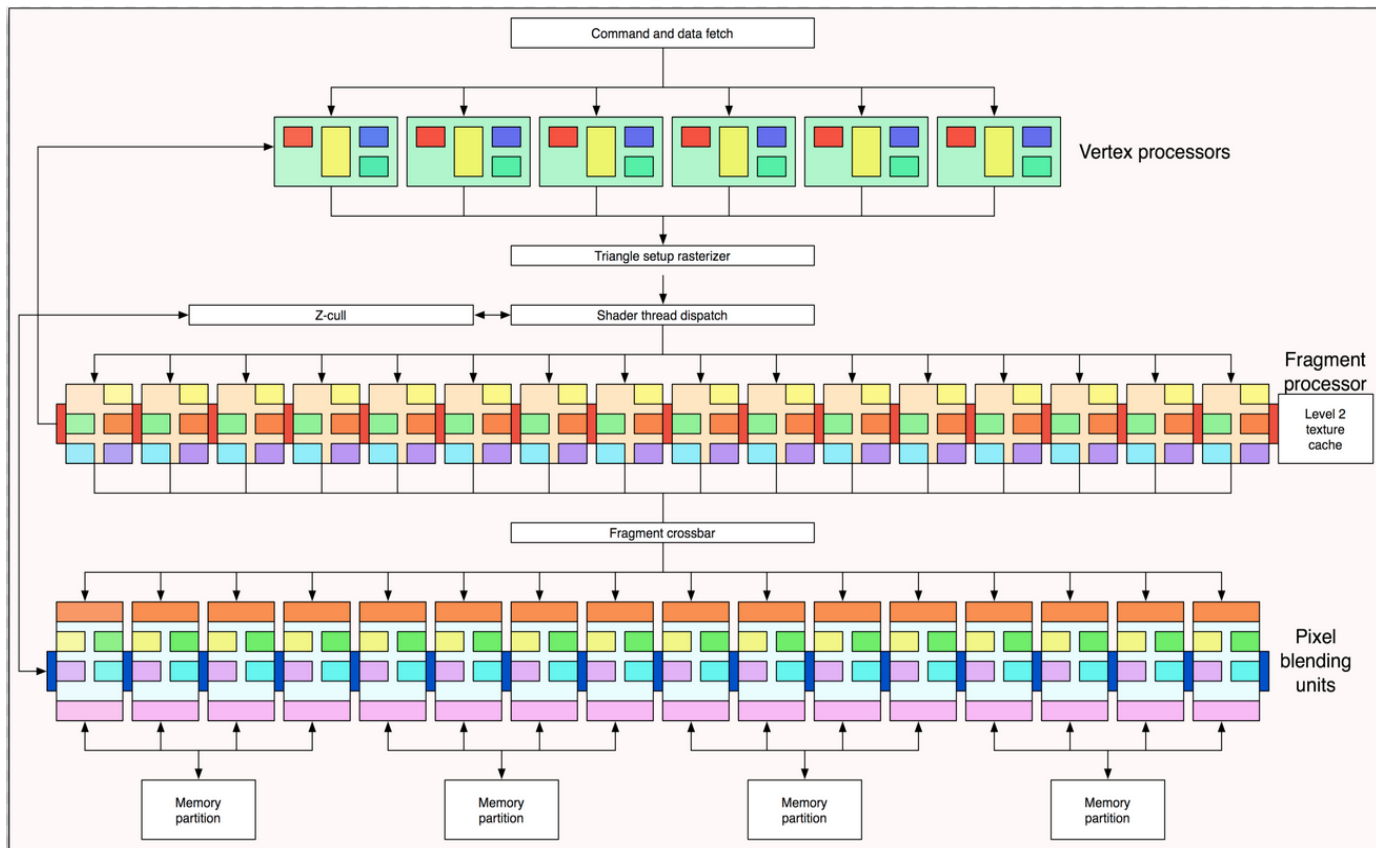
7 Carte 3D avec pixels et vertex shaders non-unifiés.

### • Architecture traditionnelle (pré-unifiée)

Les GPU possèdent :

- Un pipeline de **vertex shaders**
- Un pipeline de **pixel shaders**
- Parfois des **geometry shaders** (DirectX 10) Chaque pipeline est **spécialisé**, ce qui limite les possibilités :
- Le GPU ne peut pas redistribuer dynamiquement les ressources.
- Charge déséquilibrée entre shaders selon la scène ou l'application.

Pour donner un exemple, c'était le cas de la Geforce 6800. Elle comprenait 16 processeurs pour les *pixel shaders*, alors qu'elle n'avait que 6 processeurs pour les *vertex shaders*. D'autres cartes graphiques plus récentes utilisent encore cette stratégie, comme l'ARM Mali 400. [6]



### 8Architecture de la GeForce 6800.

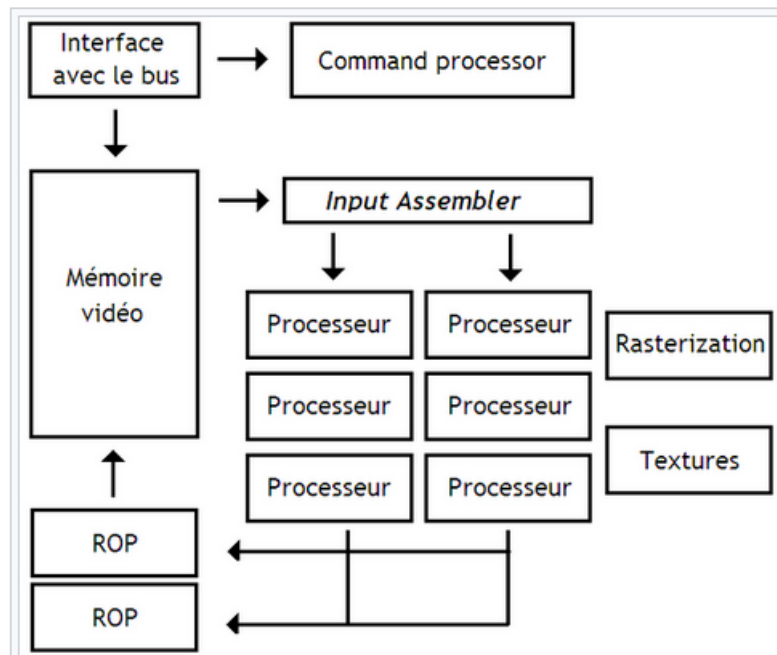
#### • Introduction des architectures unifiées (2006–2007)

Avec l'architecture unifiée (NVIDIA GeForce 8, ATI Radeon X1000) :

- Les unités de calcul deviennent **universelles (unified shader core)**.
- Toutes les unités peuvent traiter vertices, pixels, géométries, etc.
- Le GPU peut adapter dynamiquement les ressources.

L'usage de *shaders* unifiés permet d'adapter la répartition entre *vertex shaders* et *pixels shaders* suivant les besoins de l'application, là où la séparation entre unités de vertex et de pixel ne le permettait pas. [6]

Cette étape prépare le terrain pour un GPU conçu pour le calcul général, mais l'environnement logiciel reste limité.



9 Carte 3D avec pixels et vertex shaders unifiés.

## • Les premiers langages et bibliothèques GPGPU

### BrookGPU (Université Stanford)

- Première abstraction de « flot de données » pour GPU.
- Supporte opérations vectorielles massives.
- Simplifie un peu la programmation shader.
- Limite : toujours basé sur le modèle graphique.

### Sh et RapidMind

- Approche orientée objets.
- Bonne abstraction pour les calculs mathématiques.
- Limite : performances variables.

### DirectX et OpenGL comme outils scientifiques

- Utilisation détournée des API graphiques.
- API non conçue pour le HPC.
- Aucune gestion native des threads massifs.

### FFT et BLAS GPU pré-CUDA

- Faible portabilité

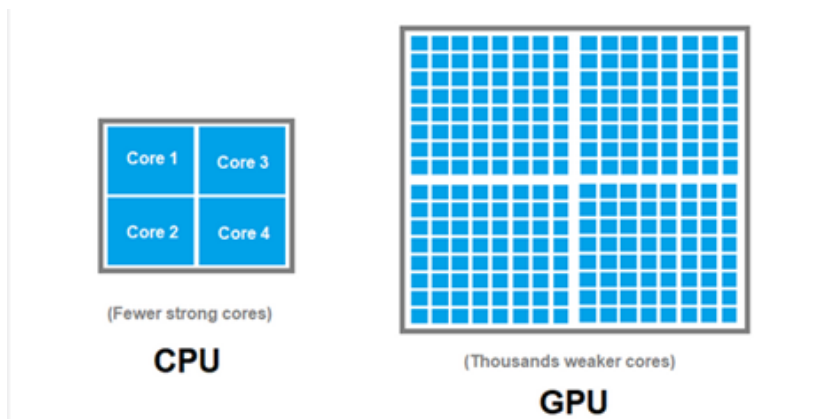
- Implémentations dépendantes du matériel
- Outils manquants pour HPC (profiling, débogage, mémoire partagée)

Même si les performances étaient prometteuses, la programmation restait une barrière majeure.

### **Promesses:**

Les GPU offraient déjà :

- Un nombre d'unités de calcul bien supérieur aux CPU
- Une bande passante mémoire plus large
- Des performances excellentes pour les opérations vectorielles massives



### **10 GPU vs CPU.**

Certains prototypes GPGPU montraient des gains :

- x10 à x30 pour des SIMD lourds
- x50 pour les convolutions et la transformation d'images
- Accélérations significatives en simulations fluides et calcul matriciel

### **Contraintes qui freinent l'adoption:**

- Modèle de programmation non adapté au scientifique
- Communication CPU–GPU lente
- Absence d'un langage généraliste
- Faible gestion des threads interactifs
- Pas de mémoire partagée par bloc
- Architecture pensée pour le rendu, pas pour le calcul



Ces contraintes créent un besoin clair : une architecture GPU délibérément conçue pour le HPC.

### Synthèse : Pourquoi Tesla était nécessaire ?

En 2007, les GPU offrent déjà un calcul parallèle très performant, mais :

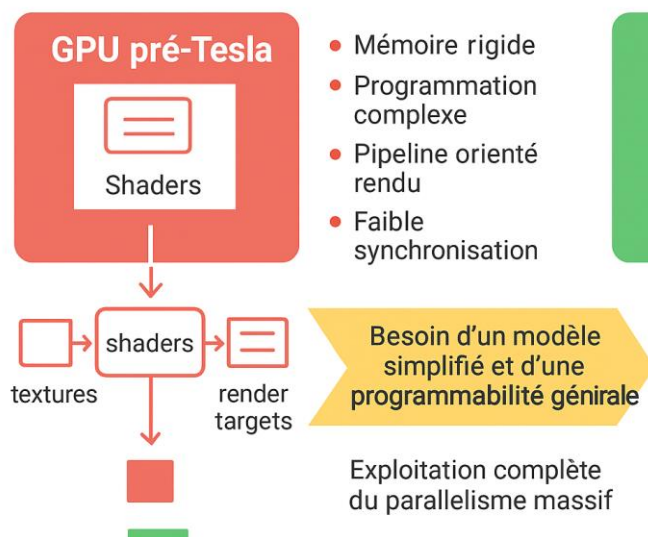
- Le modèle shader est trop compliqué.
- La gestion mémoire est trop rigide.
- Le pipeline graphique limite les algorithmes HPC.
- Les outils et langages GPGPU sont immatures.
- La programmabilité générale est quasi inexistante.

Il devient évident que :

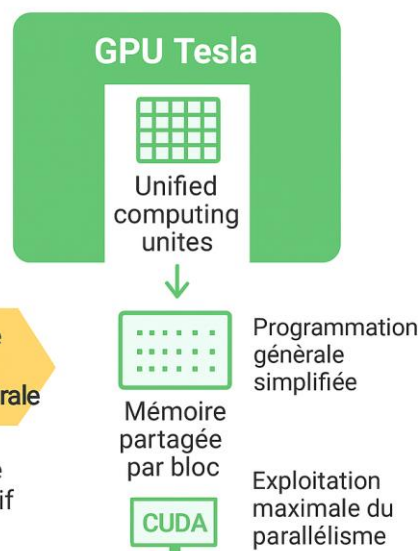
- Les GPU ont le potentiel de surpasser les CPU en calcul parallèle.
- Mais seule une refonte complète de l'architecture et du modèle de programmation peut révéler ce potentiel.

C'est dans ce contexte que naît l'architecture NVIDIA Tesla (2007), accompagnée de CUDA, qui va transformer les GPU en véritables processeurs parallèles massifs.

### **GPGPU détourné avant 2007**



### **GPU HPC dédié**



### 11 Besoin d'une architecture HPC dédiée.

## Conclusion

L'année 2007 marque l'apogée des techniques GPGPU « détournées » et la transition vers des solutions dédiées au calcul scientifique.

Malgré des performances prometteuses, les GPU pré-Tesla souffraient d'un manque d'outils adaptés, d'une architecture orientée exclusivement vers le rendu et d'une programmabilité limitée.

En décrivant les méthodes, langages et architectures existants, ce chapitre met en lumière le besoin d'un modèle unifié et simplifié, capable d'exploiter pleinement la puissance parallèle du GPU : un besoin auquel la famille Tesla répondra directement.

## **Chapitre 4 : Positionnement scientifique et industriel de Tesla.**

## Introduction

L'année 2007 marque une transformation radicale dans l'utilisation des processeurs graphiques. Après plusieurs années d'expérimentations GPGPU fondées sur le détournement du pipeline graphique, l'industrie du calcul haute performance se trouve face à un besoin clair : unifier l'architecture GPU autour du calcul parallèle général.

NVIDIA répond à cette exigence avec l'introduction de l'architecture **Tesla**, la première famille de GPU spécialement conçue pour le calcul scientifique, industriel et massivement parallèle. Ce chapitre analyse le positionnement scientifique et industriel de Tesla au moment de son lancement. Il montre comment Tesla résout les limites antérieures, comment il s'intègre dans l'écosystème du HPC, et comment il ouvre la voie à une nouvelle génération de supercalculateurs hybrides CPU–GPU.

## Contexte scientifique avant Tesla : le besoin du changement.

Au milieu des années 2000, plusieurs tendances se dégagent :

- **Saturation des CPU traditionnels**
  - Les fréquences stagnent autour de 3 GHz.
  - La loi de Dennard cesse d'être valable (limite thermique).
  - Les fabricants ajoutent des cœurs, mais le parallélisme reste limité.
- **Croissance du besoin en calcul scientifique**
  - Simulation numérique et physique
  - Calcul matriciel massif
  - Traitement d'images et vision
  - Bio-informatique
  - Apprentissage automatique (début du deep learning).

Les CPU deviennent insuffisants pour ces workloads massivement parallèles.

- **Premiers succès du GPGPU académique**

Les chercheurs montrent que :

- Un GPU peut surpasser un CPU d'un facteur  $\times 10$  à  $\times 50$
- Mais la programmation reste complexe, fragile et non portable
- Les outils manquent (débugage, profiling, mémoire partagée)

Ce contexte crée un besoin critique pour une architecture GPU dédiée au calcul, et non au rendu graphique.

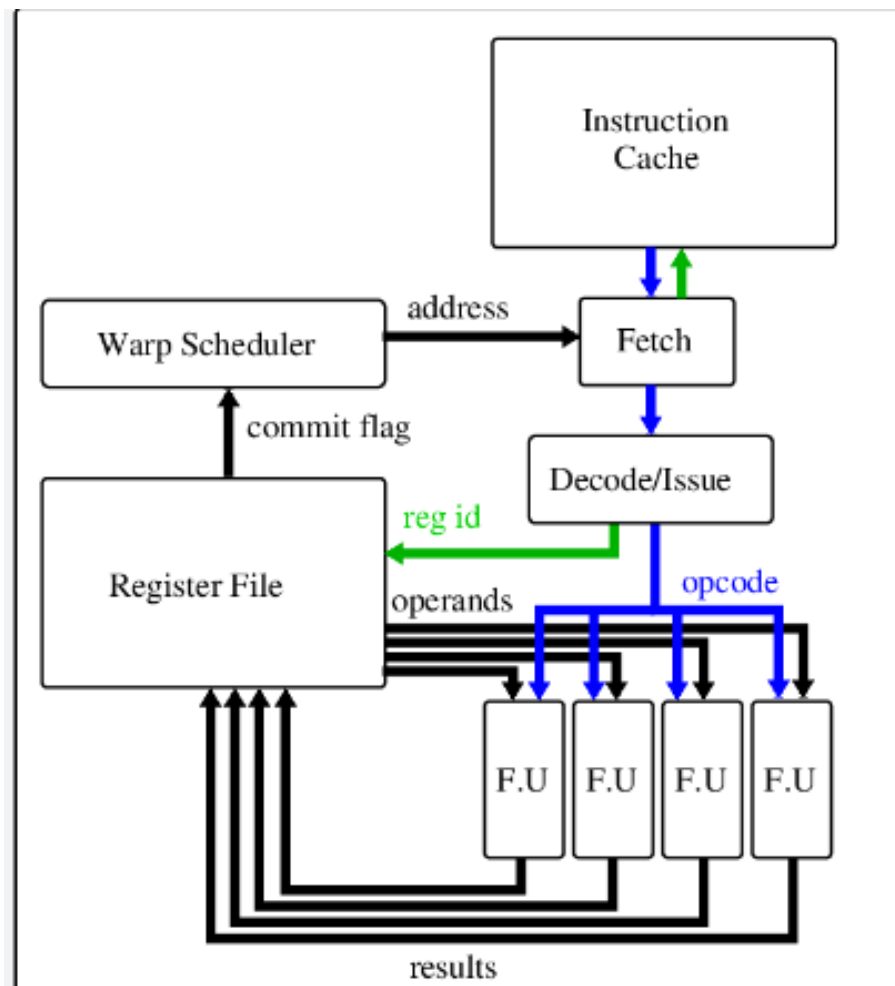
## Positionnement technologique : Tesla comme rupture architecturale.

- **Passage d'un GPU graphique à un processeur parallèle général**

Tesla est le premier GPU conçu explicitement pour :

- Le calcul scientifique
- L'exécution de threads massifs
- Le parallélisme multi-blocs
- Les opérations arithmétiques hautes performance.

Contrairement aux architectures précédentes basées sur des shaders, Tesla introduit une architecture SIMT (Single Instruction, Multiple Threads) orientée calcul. [7]



12 L'architecture SIMT[7].

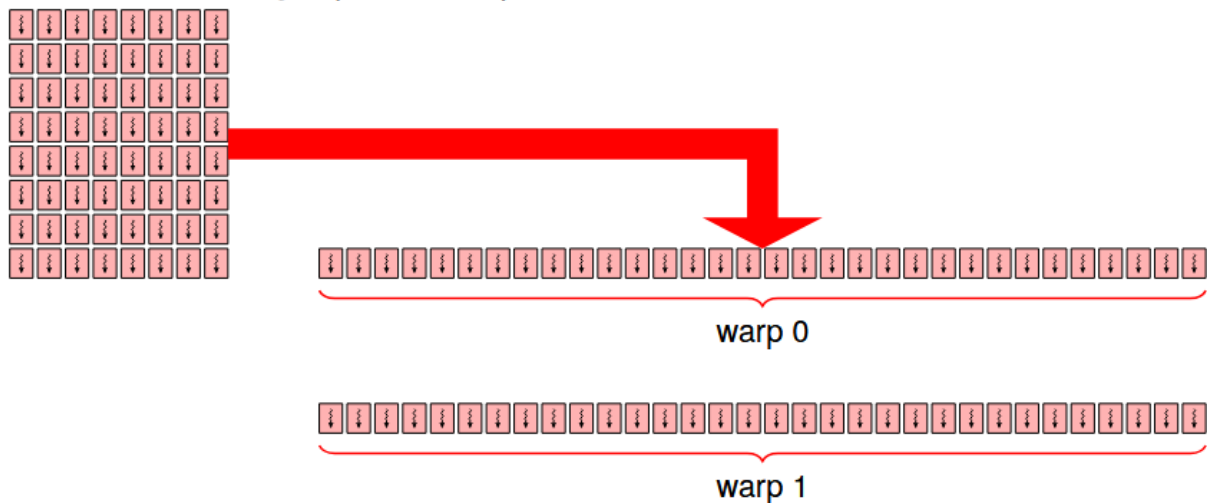
## • Clarification du modèle SIMT et de la divergence

Il est crucial de distinguer le modèle d'exécution **SIMD** (Single Instruction, Multiple Data) des CPU de l'approche **SIMT** (Single Instruction, Multiple Threads) adoptée par Tesla.

Le matériel sous-jacent du GPU fonctionne comme un processeur vectoriel (SIMD), mais l'architecture logicielle **CUDA** introduit l'abstraction SIMT. Cette abstraction permet aux programmeurs d'écrire du code comme s'il s'agissait de threads indépendants, chacun ayant son propre compteur de programme.

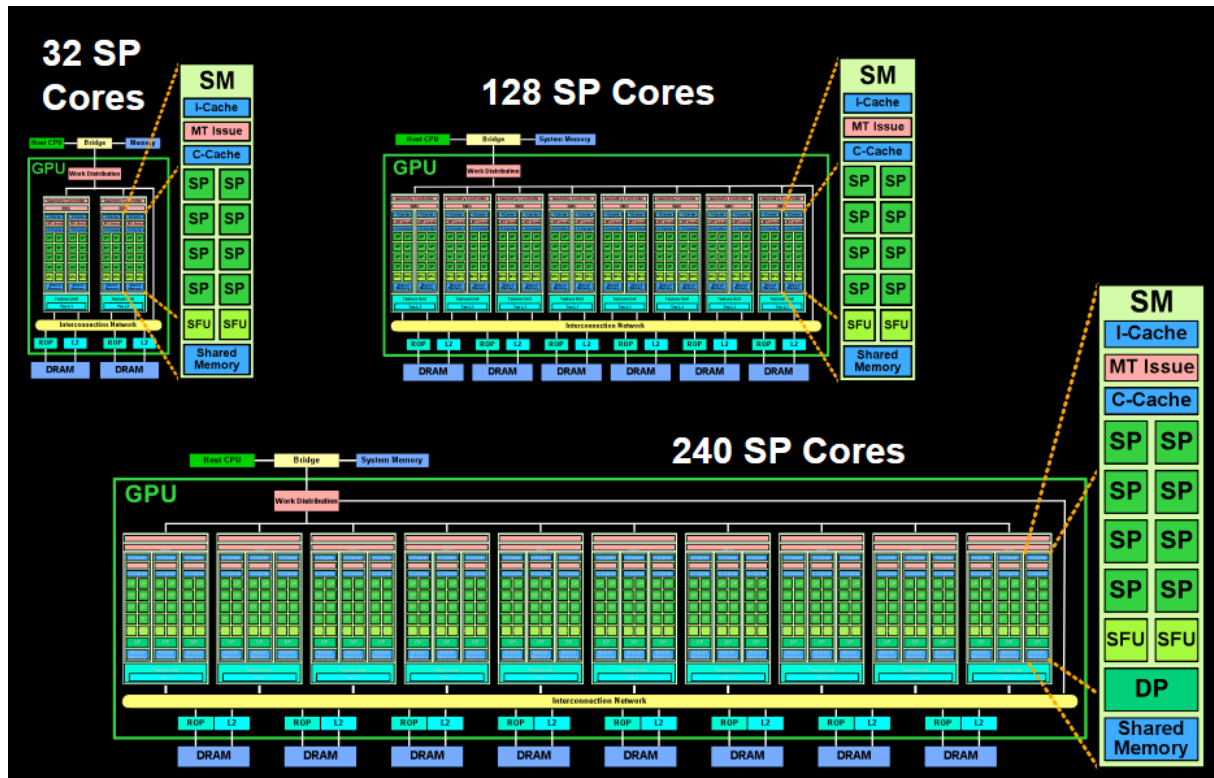
L'unité de base d'exécution est le **warp** (32 threads). En cas de branchement conditionnel (if/else) au sein d'un warp, le matériel procède à une **désactivation (prédication)** des threads divergents, forçant l'exécution séquentielle des différents chemins. [8]

Les threads sont regroupés en «warp», c-à-d 32 threads exécutées simultanément :



### 13 Comment sont exécutées les threads en CUDA

Cette gestion matérielle de la divergence est ce qui rend la programmation en CUDA plus simple que le SIMD pur, tout en conservant l'efficacité du parallélisme de données. [9]



14 La taille des GPU exige la scalabilité de CUDA.

## Introduction d'un modèle mémoire adapté au HPC

### Tesla apporte:

- Mémoire partagée rapide par multiprocesseur (SM)
- Mémoire globale adressable
- Registres massivement disponibles
- Accès mémoire coalescés. Cela répond directement aux besoins des algorithmes scientifiques.
- Unité de calcul unifiée (Streaming Multiprocessor).

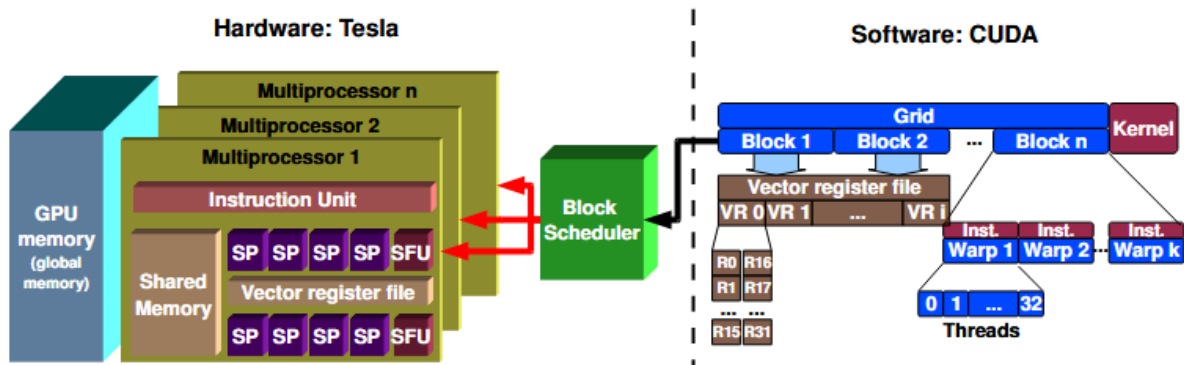
### Chaque SM de Tesla intègre:

- 8 à 32 unités de calcul
- Un ordonnanceur matériel
- Une mémoire partagée
- Un cache d'instructions

C'est la première brique d'un véritable processeur parallèle massif programmable.

## Positionnement logiciel : Tesla + CUDA = révolution programmatrice.

La réelle nouveauté de Tesla n'est pas seulement matérielle : C'est la combinaison **Tesla + CUDA**[8].



### 15 L'exécution d'un programme CUDA.

**CUDA** : un modèle de programmation généraliste.

- **CUDA (2007) permet, pour la première fois :**
  - De programmer le GPU en C/C++
  - D'abandonner les shaders
  - De gérer des milliers de threads facilement
  - De déboguer et profiler les programmes GPU
- **Adoption scientifique immédiate**

Les chercheurs adoptent CUDA car :

- Il est simple
- Il est portable
- Il offre des performances inédites
- Il permet d'exprimer tout type d'algorithme (tri, graphes, PDE, ML, etc.)
- **Premières bibliothèques scientifiques GPU**

Avec Tesla apparaissent :

- **cuFFT** (transformée de Fourier)
- **cuBLAS** (linalg)



## TESLA

- Thrust
- CUDA Math Library

Ces briques rendent Tesla immédiatement utilisable dans les laboratoires et la **carte la plus utilisée à cette époque est la NVIDIA Tesla C1060**, suivie du S1070, puis des GeForce haut de gamme (8800 GTX, GTX 280) utilisées par les chercheurs. [10]

Tesla C1060 Computing Processor	
Processor	1 x Tesla T10P
Number of cores	240
Core Clock	1.33 GHz
On-board memory	4.0 GB
Memory bandwidth	102 GB/sec peak
Memory I/O	512-bit, 800MHz GDDR3
Form factor	Full ATX: 4.736" (H) x 10.5" (L) Dual slot wide
System I/O	PCIe x16 Gen2
Typical power	160 W

16 NVIDIA Tesla C1060.

Tesla S1070 1U System	
Processors	4 x Tesla T10P
Number of cores	960
Core Clock	1.5 GHz
Performance	4 Teraflops
Total system memory	16.0 GB (4.0 GB per T10P)
Memory bandwidth	408 GB/sec peak (102 GB/sec per T10P)
Memory I/O	2048-bit, 800MHz GDDR3 (512-bit per T10P)
Form factor	1U (EIA 19" rack)
System I/O	2 PCIe x16 Gen2
Typical power	700 W

17 NVIDIA Tesla S1070.

## Positionnement industriel : l'arrivée des GPU dans les datacenters

- **Tesla : l'entrée officielle de NVIDIA dans l'industrie**

Tesla marque l'arrivée de NVIDIA dans les secteurs industriels : **supercalculateurs** et **HPC**. Les GPU Tesla sont adoptés par :

- Le **Department of Energy (DOE, USA)** → utilisation des Tesla **C2050/C2070** dans le supercalculateur **Titan**

- Les **centres de recherche européens** → Tesla **K20/K40** pour la simulation scientifique et le HPC
- Les **laboratoires de simulation numérique** → Tesla **V100** pour des calculs intensifs et modélisations complexes.

Tesla devient l'**accélérateur standard** dans de nombreux clusters HPC.

### • **Marchés industriels impactés**

Tesla est rapidement utilisé dans :

- **Énergie** (pétrole, gaz, nucléaire) → Tesla **P100/V100** pour la simulation de réservoirs pétroliers et modélisation de centrales nucléaires
- **Finance** (modèles de risque) → Tesla **K80/P100** pour les calculs de risques financiers et simulations de marché
- **Automobile** (simulation crash-test) → Tesla **V100/T4** dans les simulateurs de crash-test et optimisation de la sécurité des véhicules
- **Médecine** (imagerie 3D) → Tesla **T4/V100** pour la reconstruction rapide d'images médicales et l'analyse volumique 3D
- **Intelligence artificielle** (deep learning moderne) → Tesla **V100/A100** pour l'entraînement de réseaux neuronaux et modèles de NLP / vision par ordinateur

### • **Positionnement stratégique de NVIDIA**

Avec Tesla, NVIDIA se positionne :

- comme un acteur majeur du HPC
- comme concurrent direct d'Intel et IBM sur le calcul
- comme leader d'un nouveau paradigme : GPU Computing

C'est le début de la marque "NVIDIA Tesla" dédiée aux datacenters

### • **Réception académique et industrielle de Tesla (2007–2009)**

**Enthousiasme scientifique :**

- Accélérations exceptionnelles des simulations CFD → Tesla **C2050**
- Gains massifs en bio-informatique (BLAST, alignement génomique) → Tesla **C2070/K20**
- Calculs matriciels **jusqu'à 20× plus rapides** → Tesla **K20/K40**
- Premières expérimentations en **machine learning** (SVM, réseaux neuronaux) → Tesla **V100**

GPU PERFORMANCE COMPARISON			
	P100	V100	Ratio
Training acceleration	10 TOPS	120 TOPS	12x
Inference acceleration	21 TFLOPS	120 TOPS	6x
FP64/FP32	5/10 TFLOPS	7.5/15 TFLOPS	1.5x
HBM2 Bandwidth	720 GB/s	900 GB/s	1.2x
NVLink Bandwidth	160 GB/s	300 GB/s	1.9x
L2 Cache	4 MB	6 MB	1.5x
L1 Caches	1.3 MB	10 MB	7.7x

18 Performances des cartes Tesla P100 et V100.

## Début du GPU dans les supercalculateurs TOP500

Dès 2008–2009, les premiers supercalculateurs hybrides apparaissent. Tesla contribue à la création d'une nouvelle génération de machines hétérogènes.

- **Limites perçues à l'époque**

Malgré l'enthousiasme, certaines difficultés sont identifiées :

- **Optimisation logicielle complexe** : il était difficile de modifier les codes existants pour tirer parti des GPU.
- **Bande passante PCIe limitée** : le transfert de données CPU ↔ GPU était un goulot d'étranglement.
- **Mémoire non unifiée** : absence d'une mémoire partagée efficace entre CPU et GPU.
- **Maturité logicielle faible** : les bibliothèques et outils étaient encore à leurs débuts.

## Conclusion

L'architecture NVIDIA Tesla occupe une place unique dans l'histoire du calcul parallèle. Elle marque la transition entre un GPU dédié au rendu graphique et un véritable processeur parallèle général, conçu pour répondre aux besoins croissants du calcul scientifique et industriel. Grâce à l'introduction de CUDA et à son architecture SIMT, Tesla établit les fondations du GPU Computing moderne et positionne NVIDIA comme acteur central du HPC. Son impact se manifeste dès son lancement par une adoption rapide dans les centres de recherche, l'industrie et les supercalculateurs. Tesla n'est pas seulement une évolution : c'est le point de départ d'une révolution technologique et scientifique.

## **CHAPITRE 5 : Présentation générale de l'architecture Tesla.**

## Introduction

L'architecture NVIDIA Tesla, introduite en 2007 avec le GPU G80, représente un tournant majeur dans l'évolution des processeurs graphiques. Contrairement aux générations précédentes, conçues principalement pour le rendu 3D, Tesla adopte une architecture unifiée et orientée vers le calcul parallèle général. Ce chapitre présente les principes fondamentaux de cette architecture, son contexte de développement et les raisons qui en font une rupture technologique. [11]



19 GPU G80.

## Contexte de développement de la génération G80

- Limites du pipeline graphique fixe
- Explosion de la demande en calcul parallèle
- Premiers travaux GPGPU académiques
- Volonté de NVIDIA d'unifier le calcul graphique et général

Tesla marque ainsi le premier GPU entièrement pensé pour une utilisation scientifique.

## Objectifs technologiques de Tesla

- Transformer le GPU en **processeur parallèle massivement threadé**
- Introduire une architecture unifiée pour graphisme + calcul
- Fournir un modèle de programmation simple : **CUDA**
- Améliorer la flexibilité des shaders et dépasser leurs limitations

## Caractéristiques globales de l'architecture Tesla

- Architecture **unifiée**

- Première génération CUDA
- Architecture SIMT (Single Instruction, Multiple Threads)
- Introduction des Streaming Multiprocessors (SM)
- Décomposition du GPU en blocs indépendants

### Innovations majeures introduites

- **Warp scheduler** pour gérer 32 threads simultanés
- **Mémoire partagée rapide** permettant la coopération entre threads
- **Registres massivement disponibles**
- Exécution de milliers de threads simultanés

### Conclusion

Tesla s'impose comme une architecture pionnière, combinant simplicité programmatrice et performances massivement parallèles. Elle initie le modèle de calcul GPU contemporain et prépare la voie aux générations Fermi, Kepler, Maxwell et suivantes.

## **CHAPITRE 6 : Architecture matérielle interne de Tesla.**

## Introduction

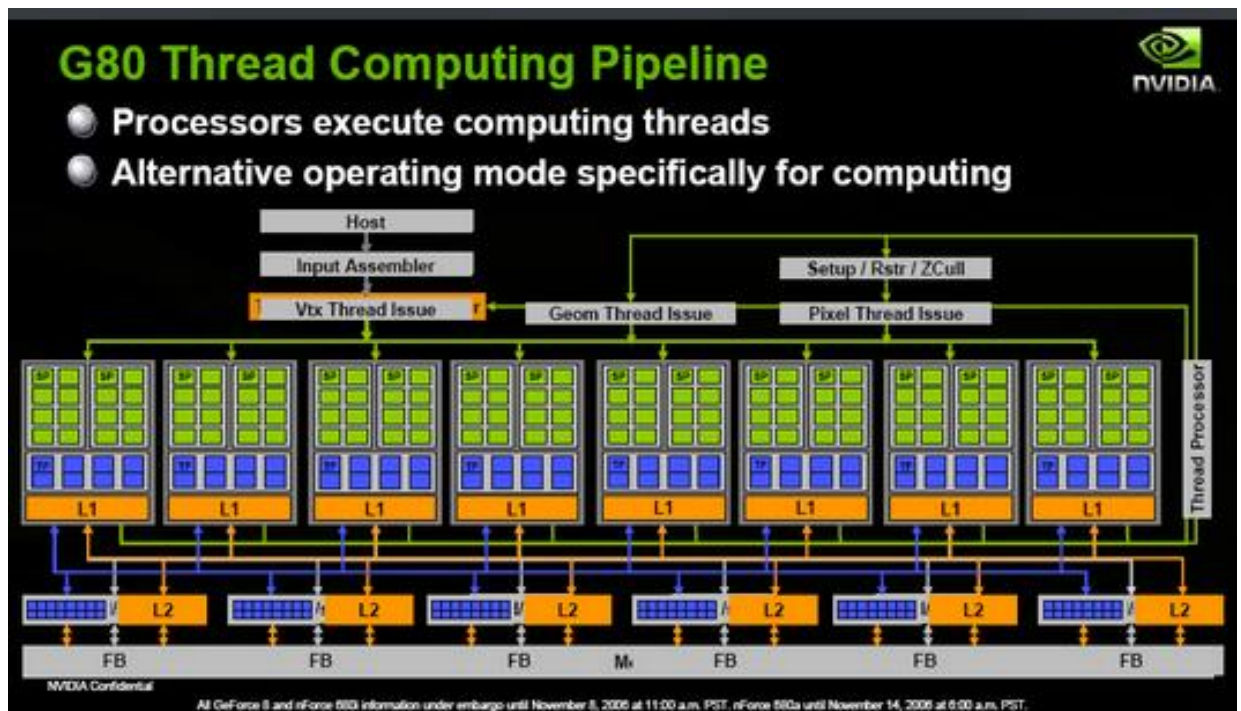
Ce chapitre étudie en détail l'organisation interne du GPU Tesla. Il analyse les composants matériels essentiels, leur rôle et la façon dont ils collaborent pour exploiter le parallélisme massif. Cette compréhension est indispensable pour optimiser les algorithmes CUDA.

## Organisation globale du GPU G80

L'architecture Tesla fut concrétisée par le processeur graphique **G80**, lancé fin 2006. Cette puce était structurée autour de **16 Streaming Multiprocessors (SM)**. Chaque SM contenait **8 Unités de Traitement Scalaires (SP)**, pour un total de **128 "CUDA Cores"** sur la puce complète. [11]

Horlogée à environ 1.35 GHz, cette architecture offrait une puissance de calcul théorique de l'ordre de **345 GFLOPS** en simple précision (FP32). Néanmoins, cette première implémentation souffrait d'une bande passante mémoire relativement limitée (environ **76.8 GB/s** sur un bus 384-bit) et d'un manque notable de hiérarchie de cache efficace, forçant une forte dépendance aux accès coalescés à la mémoire globale. [11]

- Structure hiérarchique : GPU → TPC → SM → cores
- Organisation homogène, architecture unifiée



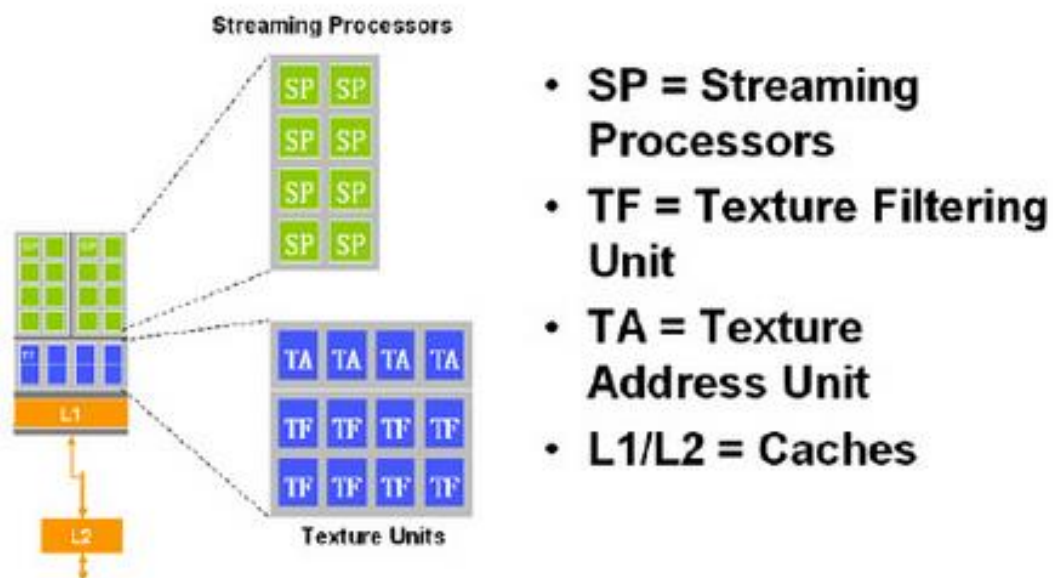
### 20 Chaîne de traitement pour le calcul parallèle des threads sur le G80.

- **Le Streaming Multiprocessor (SM)**
- 8 Streaming Processors (SP) = unités scalaires
- 1 Multi-thread Scheduler



- 1 unité SFU (Special Function Unit)
- Mémoire partagée 16 KB
- Registres 8 000 par SM Le SM constitue l' **unité fondamentale d'exécution parallèle**. [11]

## Streaming Processors, Texture Units, and On-chip Caches



### 21 L'architecture parallèle du G80 : MIMD 8 voies × SIMD 16.

- **Organisation des threads, warps et blocs**
  - 1 warp = 32 threads exécutés en SIMT
  - Un bloc = ensemble de warps exécutés sur le même SM
  - Un grid = ensemble des blocs d'un kernel
  - Gestion matérielle des contextes, latences et scheduling
- **Pipeline d'exécution**
  - Multi-issue pipeline
  - Exécution SIMT → divergence pénalisante
  - Instructions scalaires, pas vectorielles
  - Gestion des branches via masquage

- **Unités fonctionnelles**

- Unités ALU pour opérations arithmétiques
- SFU pour trigonométrie/logarithmes
- Load/Store Units pour mémoire globale
- Texture units encore disponibles mais secondaires

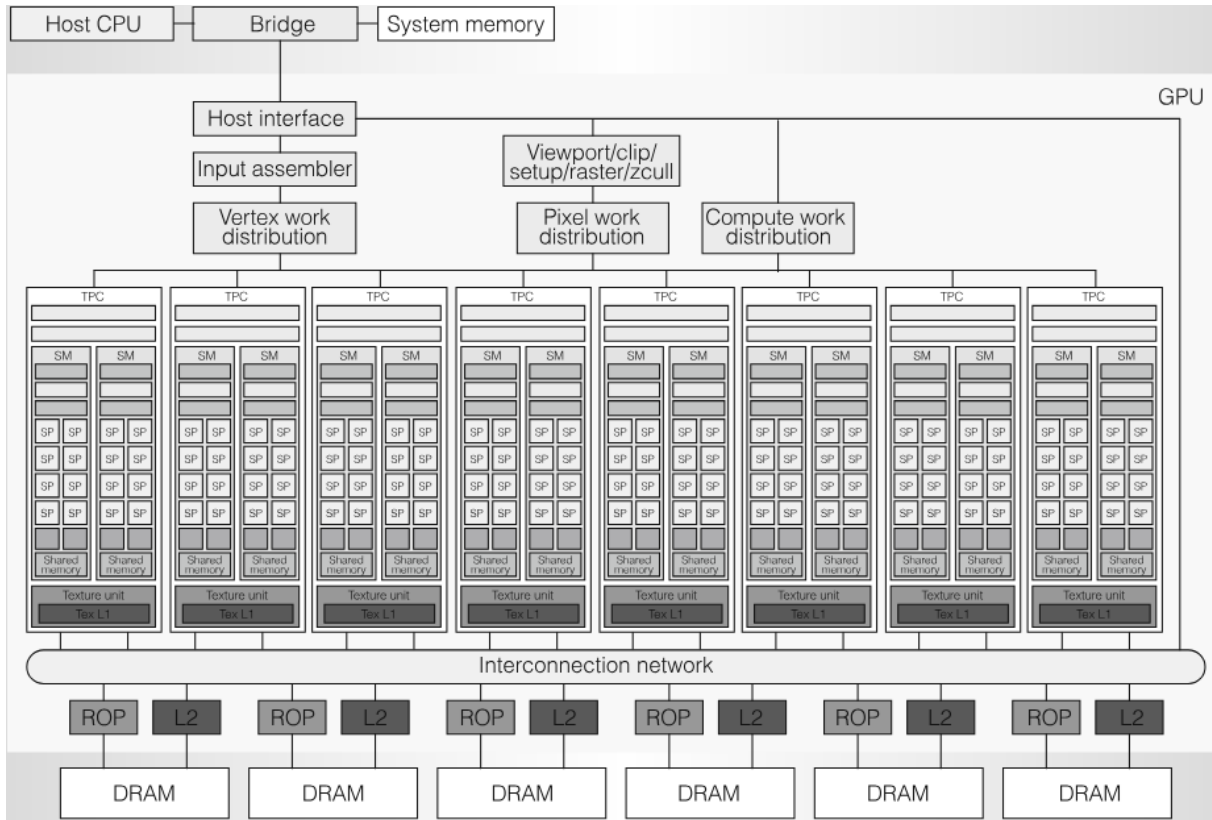
## Conclusion

Tesla pose les bases d'une architecture interne simple mais puissante, optimisée pour l'exécution massive de threads<sup>319</sup>. Le SM devient l'élément central du calcul parallèle sur GPU<sup>320</sup>.

## CHAPITRE 7 : Hiérarchie mémoire de Tesla.

## Introduction

La hiérarchie mémoire est l'un des aspects les plus déterminants pour les performances sur Tesla. Ce chapitre expose les différents niveaux de mémoire et explique leurs caractéristiques, leurs latences et leurs rôles dans l'exécution des kernels CUDA. [1]



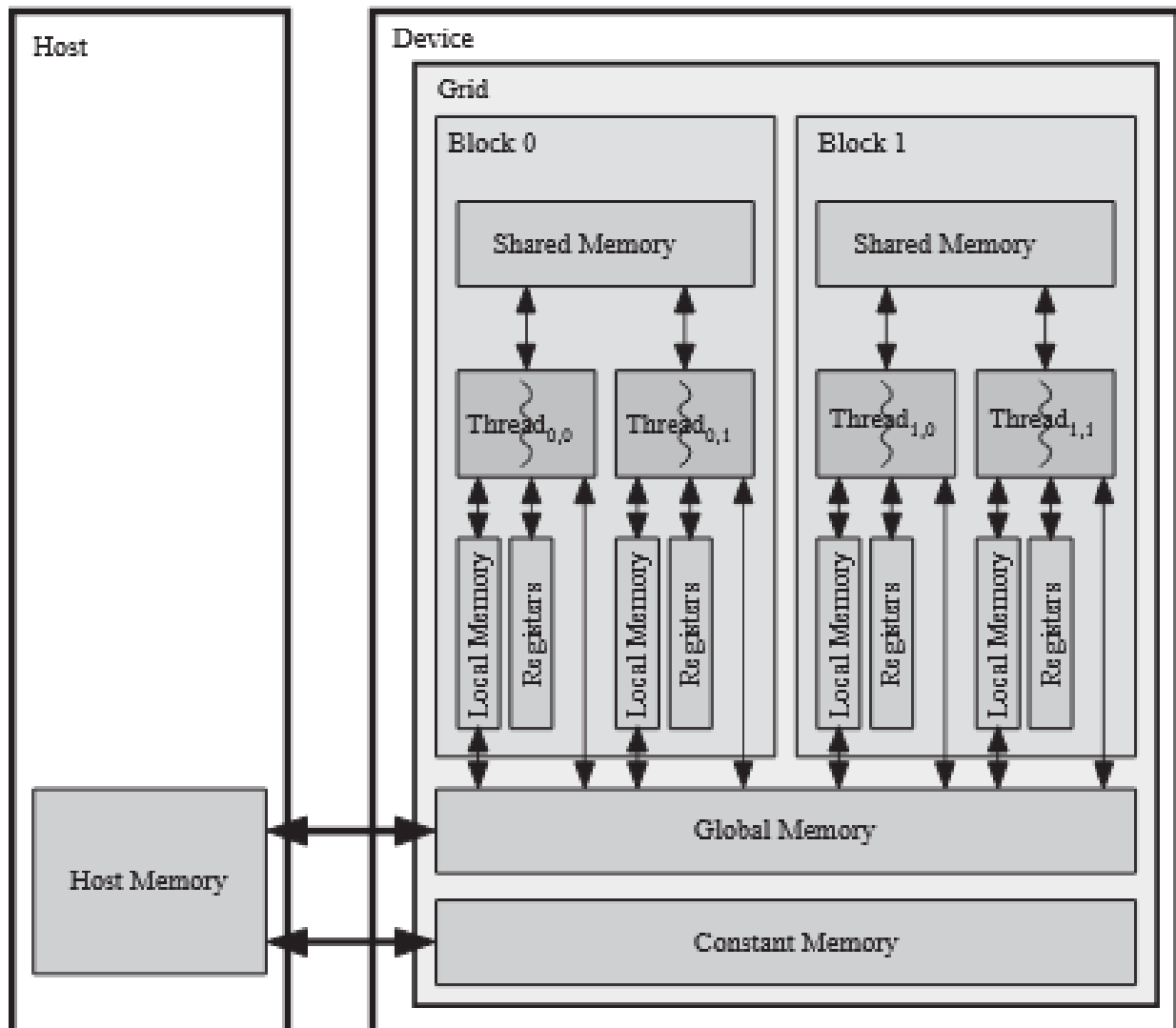
**22 L'architecture unifiée GPU Tesla pour le graphisme et le calcul.**

## Mémoire globale

- Grande capacité (~768 MB)
- Latence élevée (~400–600 cycles)
- Accès coalescés obligatoires pour de bonnes performances
- Non caché

## Mémoire partagée (Shared Memory)

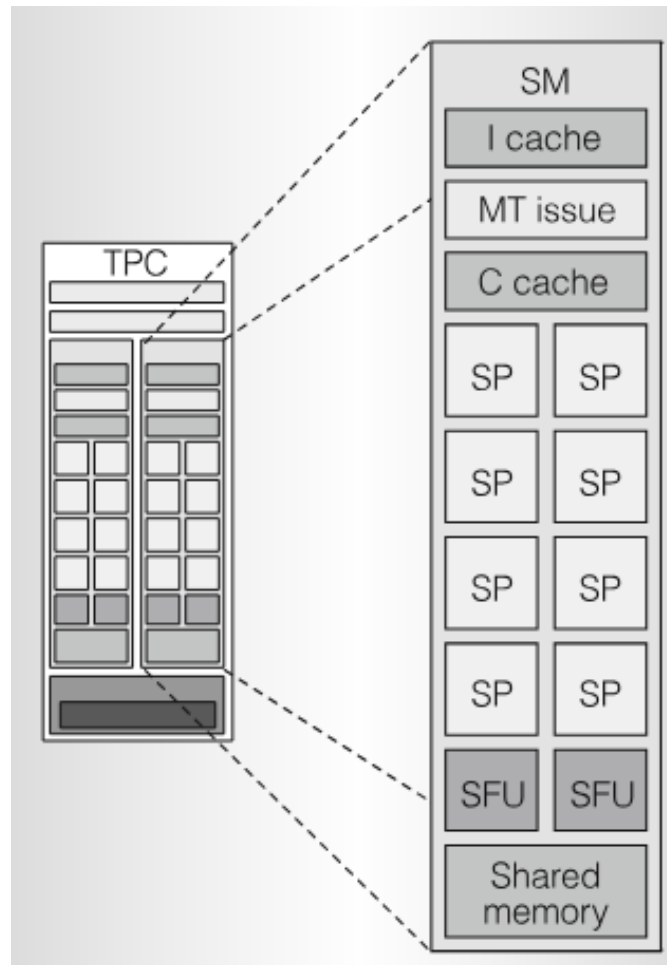
- 16 KB par SM[8]
- Faible latence (~2–4 cycles)
- Permet la coopération entre threads d'un bloc
- Organisation en banques → risques de conflits
- C'est l'élément majeur d'optimisation sur Tesla



23 La communication hôte - «device» [8].

## Registres

- 8 000 registres par SM [1]
- Très faible latence
- Utilisés par chaque thread individuellement
- Leur quantité limite l'occupation (occupancy).



24 Multiprocesseur de flux (SM) dans le GPU Tesla.

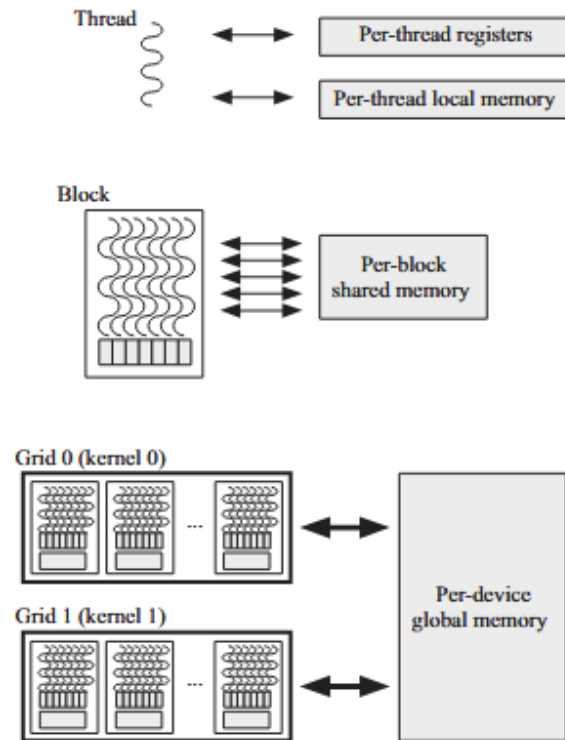
## Mémoire constante et texture

- Mémoire constante : lecture rapide si bien utilisée
- Texture : optimisée pour accès 2D, interpolation
- Utilisées pour contourner les limites de la mémoire globale

## Bande passante, cohérence et limitations

- Bande passante limitée pour le HPC moderne
- Les premières générations (G80, GT200) n'avaient pas de caches L1/L2.
- PCIe → transfert CPU-GPU lent
- Importance de placer les données dans la mémoire la plus proche

## La hiérarchie mémoire



### 25 La hiérarchie de mémoire et de threads [8].

## Conclusion

Tesla possède une hiérarchie mémoire simple mais efficace, bien adaptée au calcul parallèle. Sa maîtrise constitue la clé de l'optimisation CUDA.

## CHAPITRE 8 : Limites techniques de Tesla.



## Introduction

Bien que révolutionnaire, Tesla reste une architecture **de première génération CUDA**, encore marquée par des contraintes matérielles et logicielles héritées du GPU graphique. Ce chapitre analyse les principales limites qui ont freiné les performances et l'adoption massive des premières cartes Tesla (G80, GT200).

## Divergence des warps

La programmation CUDA repose sur un modèle **SIMD élargi** où un warp de 32 threads exécute la même instruction :

- Les branchements conditionnels provoquent une **divergence des warps**, ralentissant l'exécution.
- Les applications **irrégulières** (graphes, conditions complexes, adaptatives) subissent une perte significative de performance.
- Les premières générations Tesla ne disposent pas de mécanismes avancés de réduction de divergence.

## Limitations mémoire

- **Absence de mémoire cache (L1/L2) sur G80 et GT200**

Contrairement aux CPU, les premiers GPU Tesla n'avaient :

- **ni cache L1,**
- **ni cache L2,**
- ce qui rendait les accès mémoire globale **extrêmement coûteux**.

La performance reposait entièrement sur :

- des **accès coalescés**,
- une bonne utilisation de la **mémoire partagée**, limitée à **16 KB** par SM.
- **Bande passante et latence**
  - La mémoire DRAM avait une latence élevée.
  - Le GPU dépendait fortement du **PCIe**, très lent pour les échanges CPU-GPU.
  - Pas de **mémoire unifiée** ni de transfert asynchrone avancé → surcharge mémoire.

## Limitations du modèle CUDA 1.0

Les premiers outils CUDA étaient encore immatures :

- **Absence de synchronisation inter-blocs**, limitant fortement certains algorithmes parallèles.
- **Pas de mémoire unifiée** : le programmeur devait gérer manuellement les transferts.
- **Profiling et debugging limités**, ce qui rendait l'optimisation difficile.

- Pas encore de bibliothèques matures (cuBLAS, cuFFT, cuRAND étaient en versions initiales).

## Consommation énergétique et thermique

Les premiers GPU Tesla n'étaient pas conçus pour les datacenters :

- Architecture encore orientée **graphismes**, pas HPC.
- Gestion énergétique rudimentaire → forte consommation et émissions thermiques.
- Impact sur les grandes installations HPC :
  - densité faible,
  - besoin de refroidissement important,
  - coût énergétique élevé.

## Conclusion

Tesla a marqué une rupture technologique majeure, mais ses limites structurelles ont ouvert la voie à des améliorations successives :

- **Fermi** : introduction des caches L1/L2, mémoire ECC
- **Kepler** : efficacité énergétique
- **Pascal** : NVLink, HBM2
- **Volta / Ampere** : Tensor Cores et HPC de nouvelle génération

Ainsi, Tesla représente le **point de départ historique** du calcul GPU moderne, mais aussi un socle qui a permis d'identifier les axes clés d'évolution des générations suivantes. [12]

Device	Tesla GTX280	Fermi GTX560Ti	Kepler GTX780	Maxwell GTX980
Compute capability	1.3	2.1	3.5	5.2
SMs * cores per SM	30 * 8	8 * 48	12 * 192	16 * 128
Global memory				
Cache mechanism	N/A	L1 and L2	L2, or read-only	L2, or unified L1
Cache size	N/A	L1: 16/48 KB L2: 512 KB	Read-only: 12 KB L2: 1.5 MB	Unified L1: 24 KB L2: 2 MB
Total size	1024 MB	1024 MB	3072 MB	4096 MB
Shared memory				
Size per SM	16 KB	48/16 KB	48/32/16 KB	96 KB
Maximum size per CTA	16 KB	48 KB		
Bank No.	16	32		
Bank width	4 B		8 B	4 B
Texture memory				
Texture units	per-TPC	per-SM		
L1 cache size	6-8 KB	12 KB	12 KB	24 KB

### 26 Comparaison des hiérarchies mémoire et des capacités des GPU Tesla, Fermi, Kepler et Maxwell. [12]

## **CHAPITRE 9 : Applications majeures basées sur l'architecture Tesla.**

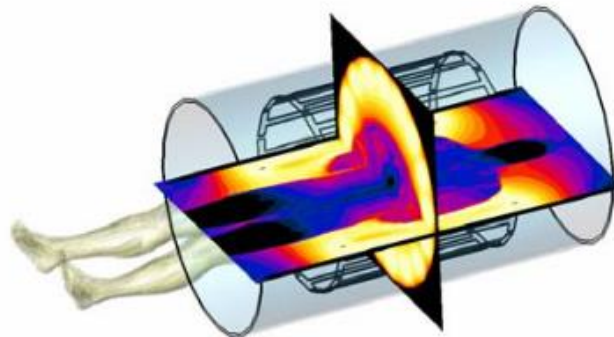
## Introduction

L'apparition de l'architecture NVIDIA Tesla en 2007 a marqué une rupture majeure dans l'utilisation des GPU au-delà du graphisme. Sa capacité à exécuter des milliers de threads parallèles en a fait une plateforme idéale pour de nombreuses applications à forte intensité de calcul. Ce chapitre présente les domaines dans lesquels Tesla a été massivement adoptée, en montrant comment ses caractéristiques matérielles et son modèle de programmation CUDA ont accéléré des algorithmes auparavant limités par les performances des CPU traditionnels.

## Simulation numérique et physique computationnelle

Tesla a permis d'accélérer de manière spectaculaire les simulations scientifiques reposant sur des modèles mathématiques complexes :

- Équations aux dérivées partielles (PDE)
- Simulation de fluides (CFD)
- Méthode des éléments finis (FEM)
- Simulation particulaire (SPH)
- Simulation électromagnétique (FDTD)



27 Simulation Boston Scientific : distribution du champ électrique à 64 MHz. [13]

### Pourquoi Tesla est efficace :

- Parallélisme massif pour calculs élémentaires indépendants
- Grande performance en virgule flottante simple précision
- Shared Memory utilisable pour stocker des zones locales de simulation

## Calcul scientifique et HPC

Tesla a été intégrée dans les clusters HPC pour :

- Matrice dense (BLAS, GEMM)

- Algèbre linéaire distribuée
- Calcul Monte-Carlo
- Modèles stochastiques et optimisation numérique

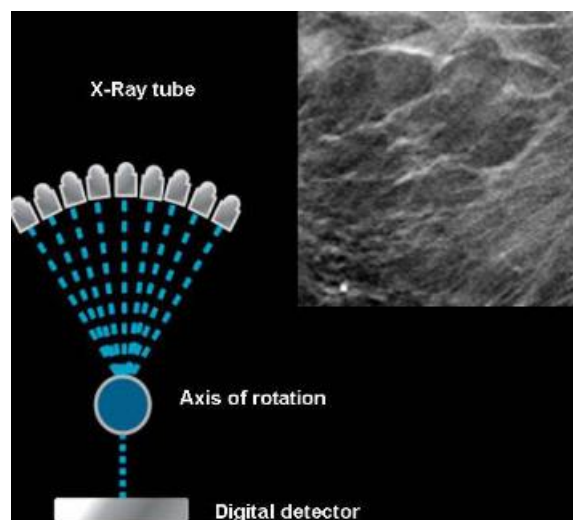
Certains centres ont reporté des accélérations  $\times 10$  à  $\times 40$  en substituant les CPU par des GPU Tesla pour les mêmes workloads.

## Traitement d'images et vision par ordinateur

Les premières bibliothèques GPGPU ont utilisé Tesla pour :

- Filtrage spatial et convolution
- Détection de contours, Sobel, Canny
- Reconstruction 3D
- Correction d'image médicale
- Segmentation et classification

Tesla a notamment été adopté dans l'imagerie médicale (scanner, IRM), où le GPU permet de reconstruire en quelques secondes ce qui prenait plusieurs minutes sur CPU.



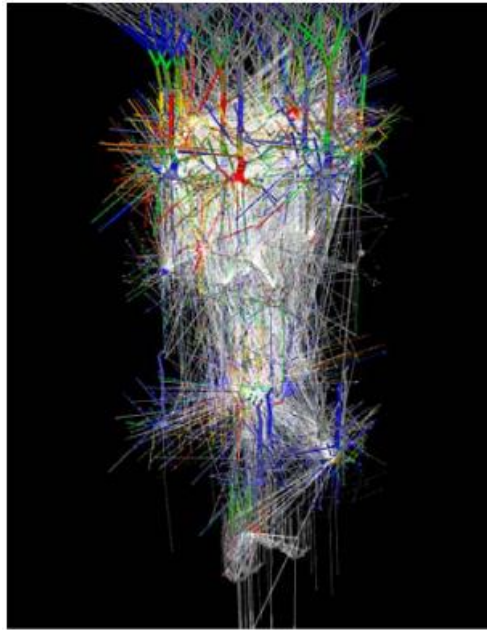
28 Imagerie médicale : tomosynthèse numérique. [13]

## Apprentissage automatique (ML) et premiers usages en IA

Même si Tesla précède l'explosion du deep learning, elle a joué un rôle majeur dans les premières approches :

- Perceptrons multicouches
- SVM accélérés sur GPU
- Réseaux de neurones primitifs pour reconnaissance d'image

Plus tard, des chercheurs (ex : Alex Krizhevsky) utiliseront les GPU pour former des réseaux profonds, en s'inspirant de l'architecture Tesla pour optimiser leurs algorithmes.



29 Simulation de circuits neuronaux : machines évoluées. [13]

## **Informatique graphique et rendu avancé**

Tesla reste un GPU et conserve des unités de traitement graphique utiles pour :

- Rendu temps réel
- Ray tracing partiel
- Simulation d'éclairage
- Calcul de shaders programmables

Le modèle unifié permet d'utiliser les unités pour calcul ou graphisme indistinctement.

## **Conclusion**

Tesla a joué un rôle central dans l'émergence du calcul parallèle moderne. Ses applications couvrent la simulation, la science, l'image, l'industrie et les premières approches d'apprentissage automatique, démontrant l'universalité de son modèle GPGPU. Elle pose les fondations des architectures qui permettront ensuite l'essor de l'intelligence artificielle moderne.

## **CHAPITRE 10 : Impact et héritage de l'architecture Tesla.**

## Introduction

Au-delà de ses performances immédiates, Tesla a profondément influencé l'évolution des architectures GPU et le paysage du calcul haut performance. Ce chapitre analyse les apports technologiques durables de Tesla, ainsi que son influence sur les générations suivantes de GPU et sur l'écosystème scientifique et industriel.

## Contribution à l'évolution du GPGPU

### Tesla marque le passage :

- d'un GPU orienté graphisme
- vers un processeur parallèle généraliste dédié HPC

### Ses contributions majeures :

- Architecture unifiée : une seule famille d'unités pour tout type de calcul
- Introduction du modèle SIMT
- Programmation simplifiée via CUDA
- Hiérarchie mémoire adaptée au calcul scientifique

Ces choix architecturaux sont aujourd'hui la norme dans les GPU modernes.

## Influence sur les générations ultérieures de GPU

Tesla a introduit les concepts fondamentaux repris ensuite par :

- **Fermi (2010)**: Ajout de cache L1 et L2, Amélioration du double précision, Plus de mémoire partagée.
- **Kepler (2012)**: Plus grande efficacité énergétique, Meilleur scheduling<sup>432</sup>.
- **Maxwell et Pascal (2015–2016)**: Optimisation des performances/Watt, Première intégration massive dans les datacenters.
- Volta, Ampere, Hopper : Introduction des Tensor Cores, Accélération massive du deep learning, NVLink haute bande passante.

Tout cela découle directement du modèle de Tesla.

## Influence sur les supercalculateurs

Grâce à Tesla :

- apparition des premiers supercalculateurs hybrides CPU+GPU
- évolution vers des dizaines de pétaFLOPS



- dominance actuelle des GPU dans TOP500
- accélération des simulations scientifiques à l'échelle mondiale

Aujourd'hui, plus de 90 % des supercalculateurs exascale utilisent des GPU héritiers de Tesla.

## **Impact sur l'intelligence artificielle moderne**

### **Sans Tesla et CUDA :**

- pas de deep learning rapide
- pas d'entraînement de réseaux complexes
- pas de modèles modernes de vision ou de langage

Tesla est souvent qualifiée de "point zéro du deep learning moderne".

## **Impact industriel durable**

Tesla a :

- transformé les workflows industriels
- accéléré la recherche pharmaceutique
- permis des systèmes embarqués avancés
- participé à la montée de l'IA dans la robotique, l'automobile, la santé, etc.

Aujourd'hui encore, les bases posées par Tesla se retrouvent dans :

- les GPU NVIDIA A100 (Ampere)
- les H100 et H200 (Hopper)
- les architectures MI de AMD
- les accélérateurs IA dédiés (TPU, IPU, etc.)

## **Conclusion**

L'architecture Tesla constitue un jalon historique dans l'évolution du calcul parallèle. Son héritage dépasse largement son époque, influençant les supercalculateurs, l'IA, la simulation scientifique et le matériel moderne. Elle représente l'architecture fondatrice du GPGPU tel qu'il existe aujourd'hui.

## Conclusion générale

Au terme de cette étude, il apparaît clairement que l'architecture **NVIDIA Tesla (G80)** occupe une place centrale dans l'évolution des technologies de calcul moderne, tant dans le domaine scientifique que dans l'industrie. Conçue initialement pour répondre aux besoins croissants en puissance de calcul, Tesla s'est imposée comme un pilier incontournable pour les applications nécessitant un traitement massif et parallèle des données.

L'analyse de son positionnement scientifique et industriel montre que Tesla a su transformer des secteurs entiers : de la recherche HPC aux simulations physiques, en passant par l'imagerie médicale, l'intelligence artificielle et les applications industrielles à grande échelle. L'intégration de Tesla dans ces domaines a ouvert la voie à de nouvelles approches méthodologiques et à de nouveaux modèles d'apprentissage. Son impact dépasse les seuls aspects techniques, s'étendant à l'organisation du travail, à l'optimisation énergétique et au développement de nouveaux marchés.

L'héritage de l'architecture Tesla s'est ensuite concrétisé par l'émergence de générations successives. Tesla a servi de fondation solide, permettant aux architectures ultérieures comme **Fermi**, puis **Kepler** et **Ampere**, de perfectionner le modèle **CUDA** et d'adresser ses limitations initiales (décrites au Chapitre 9). Les principales améliorations héritées furent l'intégration d'un **cache L1/L2 hiérarchisé** fonctionnel et une **performance largement accrue en double précision (FP64)**, rendant le GPU véritablement compétitif en HPC. Cet héritage se traduit aujourd'hui dans les standards logiciels, les frameworks d'IA et les infrastructures de calcul qui dominent le marché.

En conclusion, l'architecture Tesla représente une **étape fondatrice** et non une simple évolution dans l'histoire du calcul haute performance. Son influence continue de se faire sentir et son rôle restera essentiel dans les années à venir, notamment avec l'expansion de l'IA, du cloud et des applications scientifiques de nouvelle génération.

## Bibliographie et webographie.

[1] Lindholm, E., Nickolls, J., Oberman, S., & Montrym, J. (2008). *NVIDIA Tesla: A Unified Graphics and Computing Architecture*. IEEE Micro, 28(2), 39–55. (Article fondateur décrivant l'architecture G80 et ses spécifications détaillées.)

[2] Garland, M., & Kirk, D. (2007). *CUDA: A New Parallel Programming Model for General-Purpose GPU Computing*. ACM SIGGRAPH 2007 courses. (Introduction au modèle de programmation CUDA et à la notion de SIMT.)

[3] Volkov, V., & Demmel, J. (2008). *Benchmarking GPUs to Tune Dense Linear Algebra*. Proceedings of the 2008 ACM/IEEE Conference on Supercomputing (SC '08). (Souvent utilisé pour illustrer les premières performances et les limites du G80 en bande passante et en double précision.)

[4] Quelle est la plus ancienne carte graphique Nvidia. visiter 07/12/2025

[https://softwareg.com.au/fr-fr/blogs/materiel-informatique/quelle-est-la-plus-ancienne-carte-graphique-nvidia?srsId=AfmBOoqcSq2uxt8FtmftTrRg18R8M7RpAT06oBrohi1Nh-z\\_cq5uV4ki](https://softwareg.com.au/fr-fr/blogs/materiel-informatique/quelle-est-la-plus-ancienne-carte-graphique-nvidia?srsId=AfmBOoqcSq2uxt8FtmftTrRg18R8M7RpAT06oBrohi1Nh-z_cq5uV4ki)

[5] Inside Nvidia's GeForce 6000 Series. visiter 07/12/2025

<https://old.chipsandcheese.com/2025/04/01/inside-nvidias-geforce-6000-series/>

[6] Les architectures avec unités de *vertex* et *pixels* séparées. visiter 07/12/2025

[https://fr.wikibooks.org/wiki/Les\\_cartes\\_graphiques/La\\_r%C3%A9partition\\_du\\_travail\\_sur\\_les\\_unit%C3%A9s\\_de\\_shaders](https://fr.wikibooks.org/wiki/Les_cartes_graphiques/La_r%C3%A9partition_du_travail_sur_les_unit%C3%A9s_de_shaders)

[7] Contributions to computer arithmetic and applications to embedded systems. visited 07/12/2025 [https://www.researchgate.net/figure/Single-Instruction-Multiple-Thread-architecture\\_fig39\\_278826278](https://www.researchgate.net/figure/Single-Instruction-Multiple-Thread-architecture_fig39_278826278)

[8] Développement GPGPU visited 07/12/2025

<https://p-fb.net/master1/gpgpu/cours/cuda.pdf>

[9] Étude comparée et simulation d'algorithmes de branchements pour le GPGPU visited 07/12/2025.

<https://hal.science/hal-00397697v2/document>

[10] NVIDIA Tesla C1060 y S1070 basados en GT200 visited 07/12/2025

<https://www.madboxpc.com/nvidia-tesla-c1060-y-s1070-basados-en-gt200/>

[11] NVidia G80: Architecture visited 07/12/2025

[https://www.icare3d.org/news\\_articles/nvidia\\_g80\\_architecture.html](https://www.icare3d.org/news_articles/nvidia_g80_architecture.html)

[12] Dissecting GPU Memory Hierarchy Through Microbenchmarking visited 07/12/2025

<https://www.semanticscholar.org/paper/Dissecting-GPU-Memory-Hierarchy-Through-Mei-Chu/a170a2e07d02b1ee0281f8f657aef820cb303d4c>

[13] NVIDIA® Tesla™ GPU Computing Technical Brief visited 07/12/2025

[https://www.nvidia.com/docs/io/43395/compute\\_tech\\_brief\\_v1-0-0\\_final\\_dec07.pdf](https://www.nvidia.com/docs/io/43395/compute_tech_brief_v1-0-0_final_dec07.pdf)