

# Calcul – Cours 2: Architecture des machines parallèles

Jonathan Rouzaud-Cornabas

LIRIS / Insa de Lyon – Inria Beagle

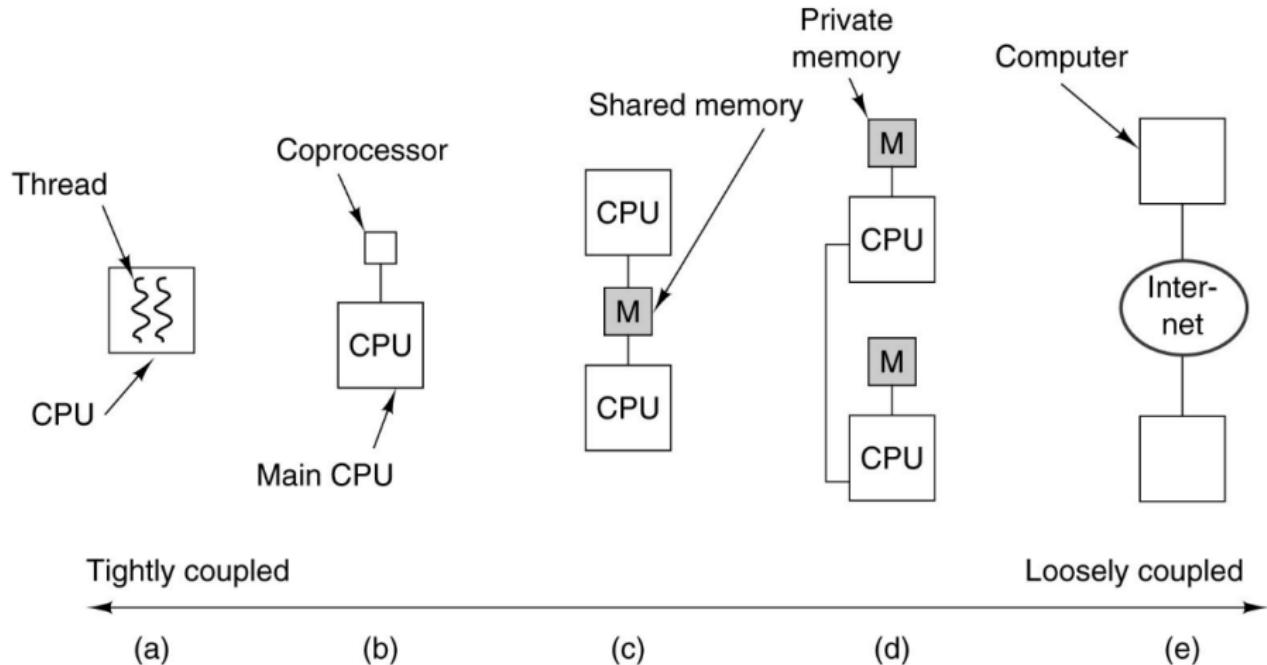
Cours inspiré de ceux de Frédéric Desprez (DR Inria – Grenoble)

# Références

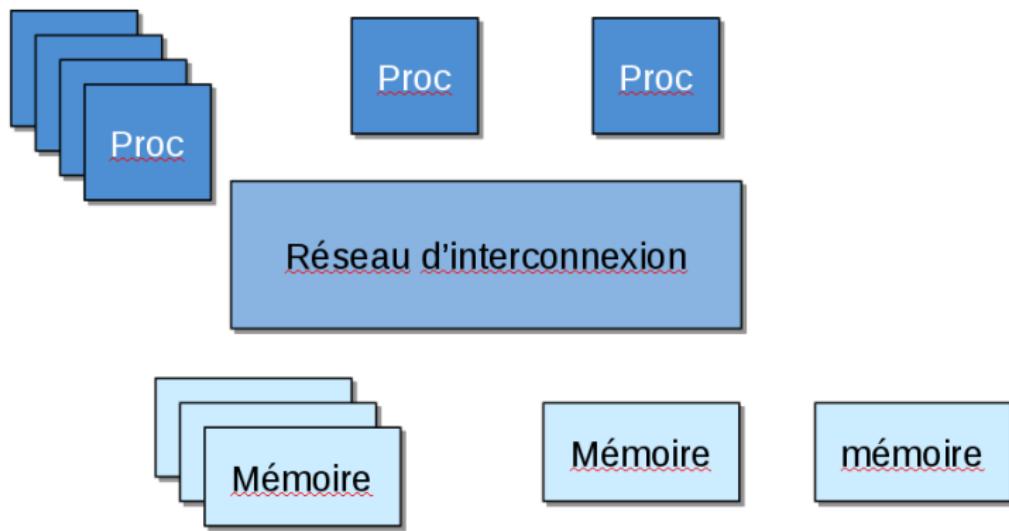
- Cours “Calcul hautes performance – architectures et modèles de programmation”, Françoise Roch, Observatoire des Sciences de l’Univers de Grenoble Mesocentre CIMENT
- 4 visions about HPC - A chat, X. Vigouroux, Bull
- J. Demmel, K. Yelick, Berkeley
- Parallel Programming – For Multicore and Cluster System, T. Rauber, G. Rünger

# Architecture de calcul

# Architectures parallèles



# Une machine parallèle générique



- Où est la mémoire ?
- Est-ce qu'elle est connectée directement avec les processeurs ?
- Quelle est la connectivité des processeurs ?

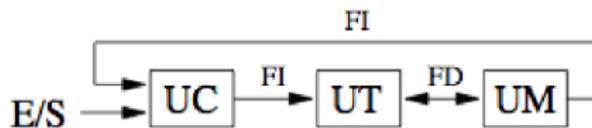
# Classification de Flynn

	Single Instruction	Multiple Instructions
Single Data	SISD	MISD
Multiple Data	SIMD	MIMD

- Caractérise les machines suivant leurs flots de données et d'instructions
- Flynn, M. (1972). "Some Computer Organizations and Their Effectiveness". IEEE Trans. Comput. C-21: 948.

# SISD: Single Instruction, Single Data stream

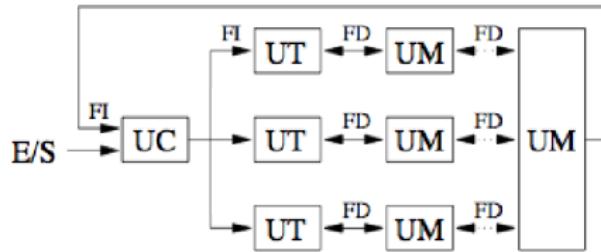
- Machines séquentielles “classiques”
- Chaque opération s’effectue sur une donnée à la fois



- UC = Unité de Contrôle (chargée du séquencage des instructions)
- UT = Unité de traitement (effectue les opérations)
- FI = Flot d'Instructions
- UM = Unité Mémoire (contient les instructions et les données)
- FD = Flot de Données
- Modèle de Von Neuman (1945)

# SIMD: Single Instruction stream, Multiple Data stream

- Unités de calcul totalement synchronisées
- Exécution conditionnelle avec flag de masquage



- Machines adaptées aux traitements très réguliers (opérations matricielles, FFT, imagerie).
- Pas adapté du tout aux opérations irrégulières.

# Conditionnelles en SIMD

- Masque d'activité : permet d'empêcher des processeurs de faire des opérations

conditional statement

```
if (B == 0)
then C = A
else C = A/B
```

initial values

<table border="1"> <tr><td>A</td><td>5</td></tr> <tr><td>B</td><td>0</td></tr> <tr><td>C</td><td>0</td></tr> </table>	A	5	B	0	C	0	<table border="1"> <tr><td>A</td><td>4</td></tr> <tr><td>B</td><td>2</td></tr> <tr><td>C</td><td>0</td></tr> </table>	A	4	B	2	C	0	<table border="1"> <tr><td>A</td><td>1</td></tr> <tr><td>B</td><td>1</td></tr> <tr><td>C</td><td>0</td></tr> </table>	A	1	B	1	C	0	<table border="1"> <tr><td>A</td><td>0</td></tr> <tr><td>B</td><td>0</td></tr> <tr><td>C</td><td>0</td></tr> </table>	A	0	B	0	C	0
A	5																										
B	0																										
C	0																										
A	4																										
B	2																										
C	0																										
A	1																										
B	1																										
C	0																										
A	0																										
B	0																										
C	0																										
Processor 0	Processor 1	Processor 2	Processor 3																								

execute  
"then" branch

<table border="1"> <tr><td>A</td><td>5</td></tr> <tr><td>B</td><td>0</td></tr> <tr><td>C</td><td>5</td></tr> </table>	A	5	B	0	C	5	<table border="1"> <tr><td>Idle</td><td></td></tr> <tr><td>A</td><td>4</td></tr> <tr><td>B</td><td>2</td></tr> <tr><td>C</td><td>0</td></tr> </table>	Idle		A	4	B	2	C	0	<table border="1"> <tr><td>Idle</td><td></td></tr> <tr><td>A</td><td>1</td></tr> <tr><td>B</td><td>1</td></tr> <tr><td>C</td><td>0</td></tr> </table>	Idle		A	1	B	1	C	0	<table border="1"> <tr><td>Idle</td><td></td></tr> <tr><td>A</td><td>0</td></tr> <tr><td>B</td><td>0</td></tr> <tr><td>C</td><td>0</td></tr> </table>	Idle		A	0	B	0	C	0
A	5																																
B	0																																
C	5																																
Idle																																	
A	4																																
B	2																																
C	0																																
Idle																																	
A	1																																
B	1																																
C	0																																
Idle																																	
A	0																																
B	0																																
C	0																																
Processor 0	Processor 1	Processor 2	Processor 3																														

execute  
"else" branch

<table border="1"> <tr><td>Idle</td><td></td></tr> <tr><td>A</td><td>5</td></tr> <tr><td>B</td><td>0</td></tr> <tr><td>C</td><td>5</td></tr> </table>	Idle		A	5	B	0	C	5	<table border="1"> <tr><td>Idle</td><td></td></tr> <tr><td>A</td><td>4</td></tr> <tr><td>B</td><td>2</td></tr> <tr><td>C</td><td>2</td></tr> </table>	Idle		A	4	B	2	C	2	<table border="1"> <tr><td>Idle</td><td></td></tr> <tr><td>A</td><td>1</td></tr> <tr><td>B</td><td>1</td></tr> <tr><td>C</td><td>1</td></tr> </table>	Idle		A	1	B	1	C	1	<table border="1"> <tr><td>Idle</td><td></td></tr> <tr><td>A</td><td>0</td></tr> <tr><td>B</td><td>0</td></tr> <tr><td>C</td><td>0</td></tr> </table>	Idle		A	0	B	0	C	0
Idle																																			
A	5																																		
B	0																																		
C	5																																		
Idle																																			
A	4																																		
B	2																																		
C	2																																		
Idle																																			
A	1																																		
B	1																																		
C	1																																		
Idle																																			
A	0																																		
B	0																																		
C	0																																		
Processor 0	Processor 1	Processor 2	Processor 3																																

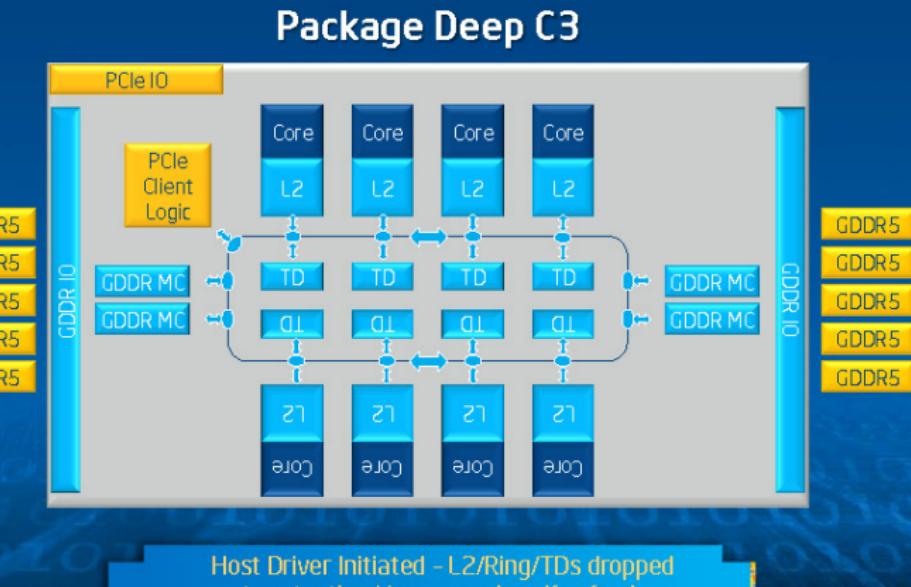
# Quelques exemples de machines SIMD

- Des machines des années 80/90
  - Illiac IV, MPP, DAC, Connection Machine CM-1/2, MasPar MP-1/2
- Un grand retour aujourd'hui
  - Processeurs Intel et mode SSE/SSE-2 (unités vectorielles)
  - NVidia Tesla: jusqu'à 2x2496 threads à 745Mhz et 12GB de RAM
  - Intel Xeon Phi: 61 cores @ 1.238Ghz et 16GB de RAM

# Quelques exemples de machines SIMD



# Quelques exemples de machines SIMD

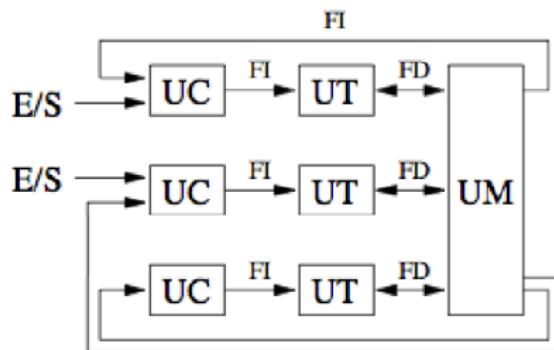


# MIMD: Multiple Instructions stream, multiple data stream

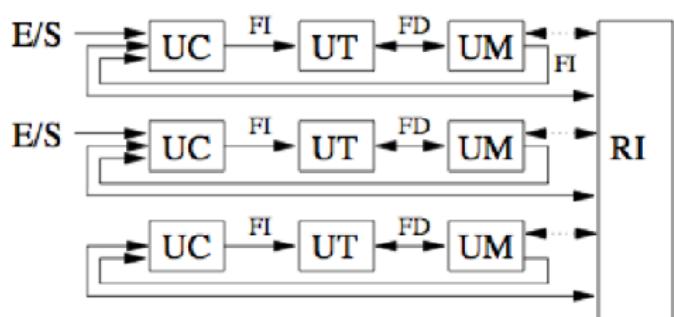
- Machines multi-processeurs
- Chaque processeur exécute son propre code de manière asynchrone et indépendante

## Deux sous-classes

Mémoire partagée



Mémoire distribuée



A mi-chemin entre SIMD et MIMD: SPMD (*Single Program, Multiple Data*)

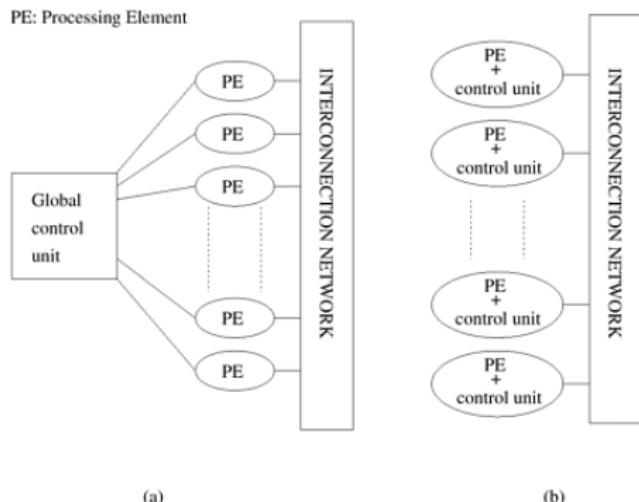
# SIMD vs MIMD

- Plates-formes SIMD

- Conçues pour des applications particulières
- Conception compliquées (et longue), pas de processeurs “sur étagère”
- Moins de matériel (une seule unité de contrôle)
- Besoin de moins de mémoire pour les instructions (un seul programme)
- Utilisé massivement pour les co-processeurs actuels

- Plates-formes MIMD

- Fonctionne pour une large variété d'applications
- Moins coûteuses (composants sur étagère, conception courte)
- Besoin de plus de mémoire (OS et programme sur chaque processeur)



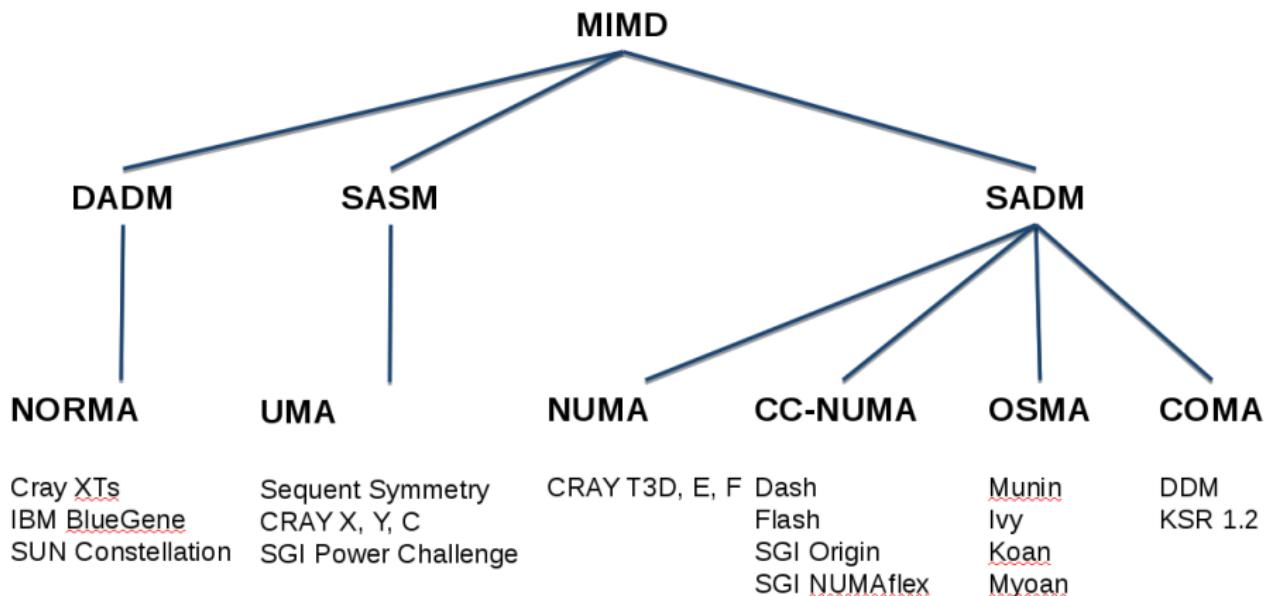
# Architecture des machines parallèles

# Classification de Raina

## Prise en compte de l'espace d'adressage

- SASM (Single Address space, Shared Memory) : mémoire partagée
- DADM (Distributed Address space, Distributed Memory) : mémoire distribuée, sans accès aux données distantes.
  - L'échange de données entre processeurs s'effectue nécessairement par passage de messages, au moyen d'un réseau de communication
- SADM (Single Address space, Distributed Memory) : mémoire distribuée, avec espace d'adressage global,
  - autorisant éventuellement l'accès aux données situées sur d'autres processeurs.

# Classification de Raina



# Modèles de programmation parallèle

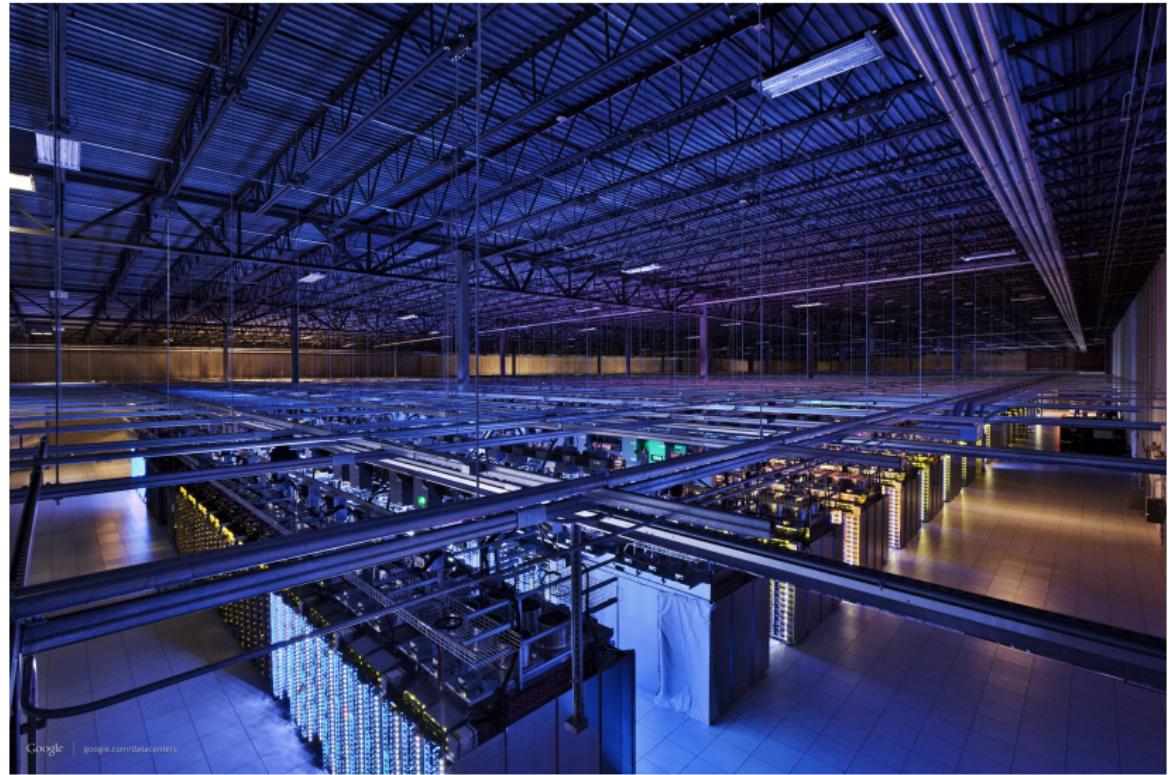
Le modèle de programmation consiste en les langages et les bibliothèques qui vont permettre d'avoir une abstraction de la machine

- Contrôle
  - Comment est créé le parallélisme (implicite ou explicite) ?
  - Quels sont les enchaînements entre les opérations (synchrones ou asynchrones) ?
- Données
  - Quelles sont les données privées et partagées ?
  - Comment sont accédées et/ou communiquées ces données ?
- Synchronisation
  - Quelles opérations peuvent être utilisées pour coordonner le parallélisme ?
  - Quelles sont les opérations atomiques (indivisibles) ?
- Coût
  - Comment est-ce que l'on peut calculer le coût de chaque élément précédent ?

# Google cluster (1997)



# Google Datacenter



Google | [google.com/datacenters](http://google.com/datacenters)

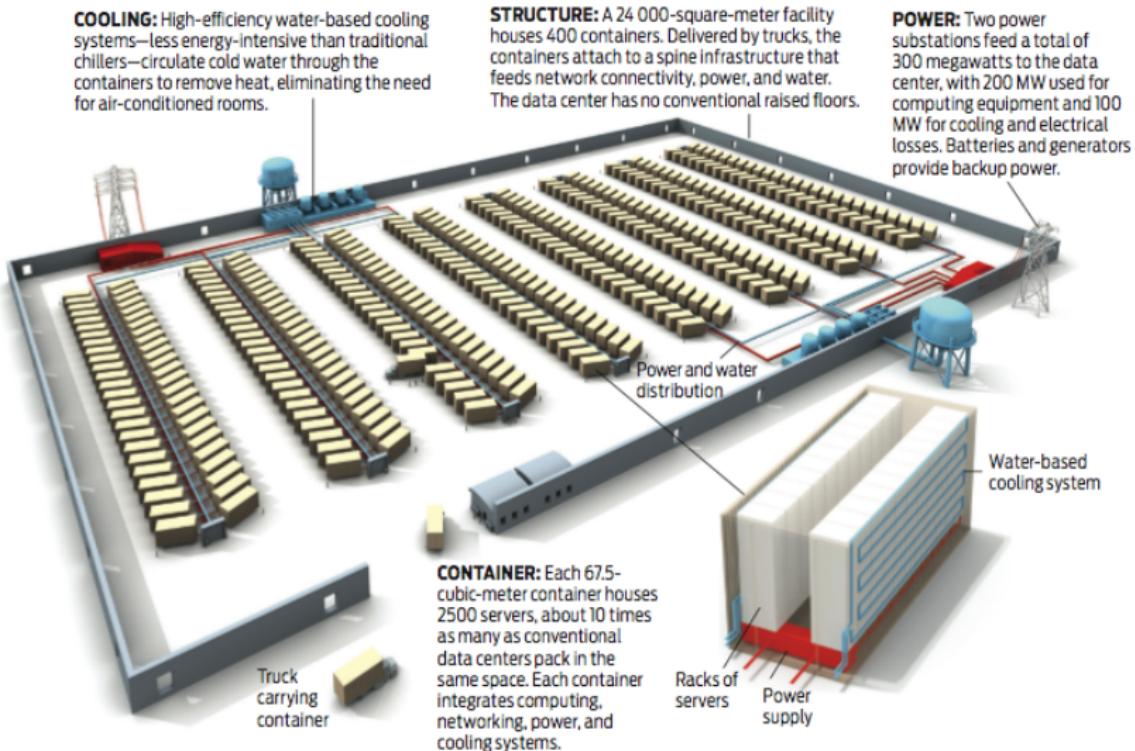
# Facebook Open Compute



# Facebook Open Compute



# Un million de serveur



# IBM RoadRunner

- Première machine à atteindre le Petaflops ( $10^{15}$  flops)
- Roadrunner contient
  - 6,948 dual-core AMD Opteron
  - 12,960 Cell engines (les mêmes que la PS3)

- 80TB de mémoire
- 288 Racks
- $560m^2$
- 10,000 connections (Infiniband et Gigabit)
- 90 km de cables



# Machines multi-cores

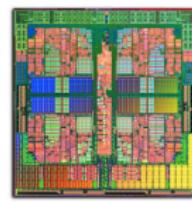
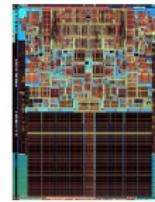
# Architectures multi-cœurs

- Un processeur composé d'au moins 2 unités centrales de calcul sur une même puce.
- Permet d'**augmenter la puissance de calcul** sans augmenter la fréquence d'horloge.
- Et donc **réduire la dissipation thermique**.
- Et **augmenter la densité** : les cœurs sont sur le même support, la connectique reliant le processeur à la carte mère ne change pas par rapport à un mono-cœur

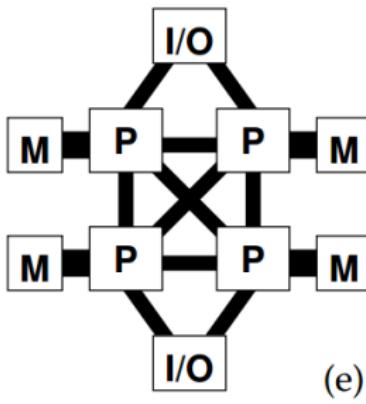
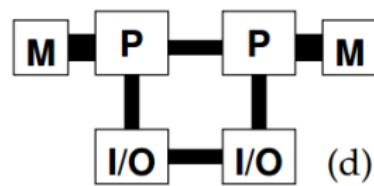
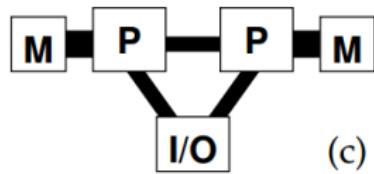
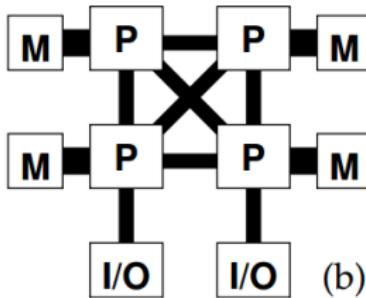
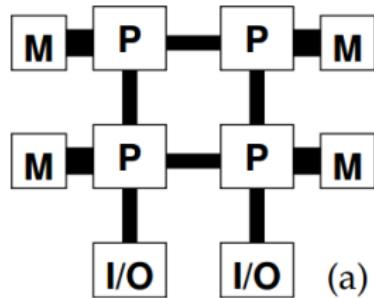
# Pourquoi les multi-cœurs ?

## Quelques ordres de grandeur

	<b>Single Core</b> Génération de gravure1	<b>Dual Core</b> Génération de gravure2	<b>Quad Core</b> Génération de gravure3
Core area	A	~ A/2	~ A/4
Core power	W	~ W/2	~ W/4
Chip power	W + O	W + O'	W + O''
Core performance	P	0.9P	0.8P
Chip performance	P	1.8 P	3.2 P



# Processeur Interconnexion



(a) four AMD Istanbul processors

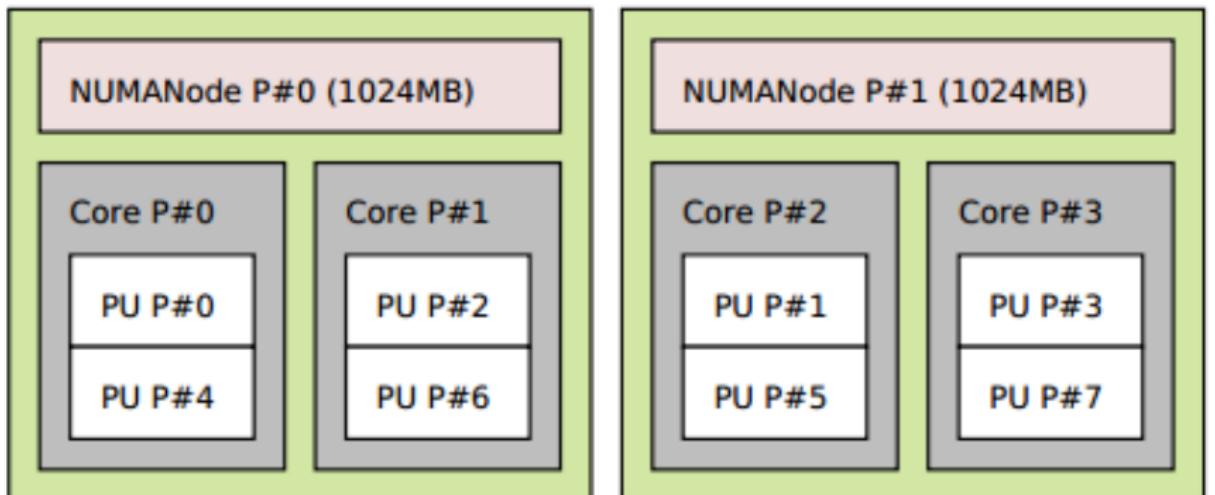
(b) four AMD Magny-Cours processors

(c) and (d) two Intel Westmere-EP processors

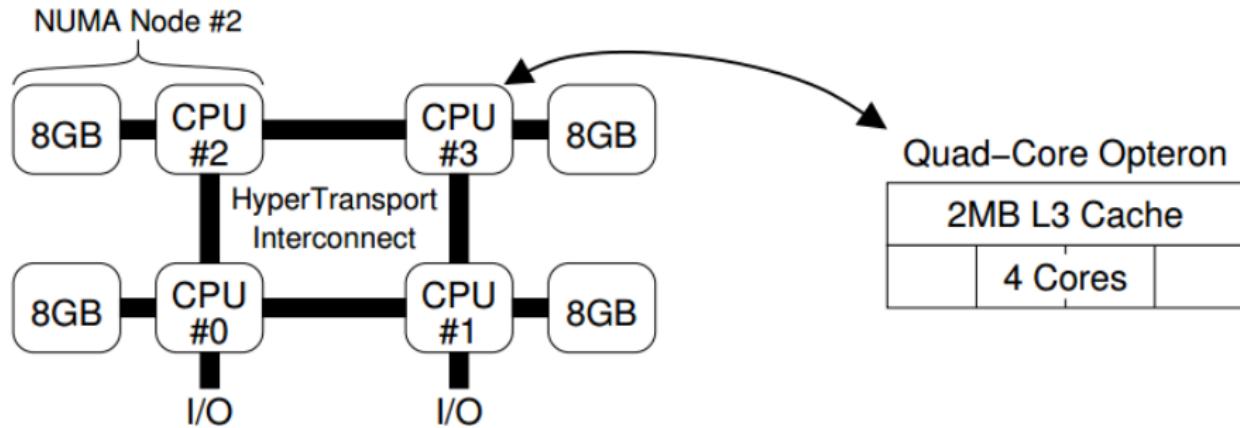
(e) four Intel Nehalem-EX processors.

# Depuis UMA vers NUMA

Machine (2048MB)

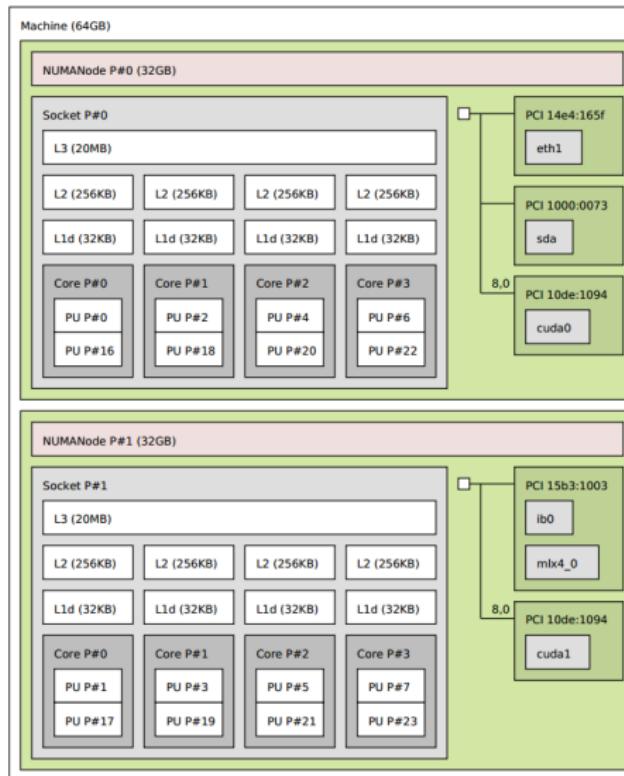


# Depuis UMA vers NUMA

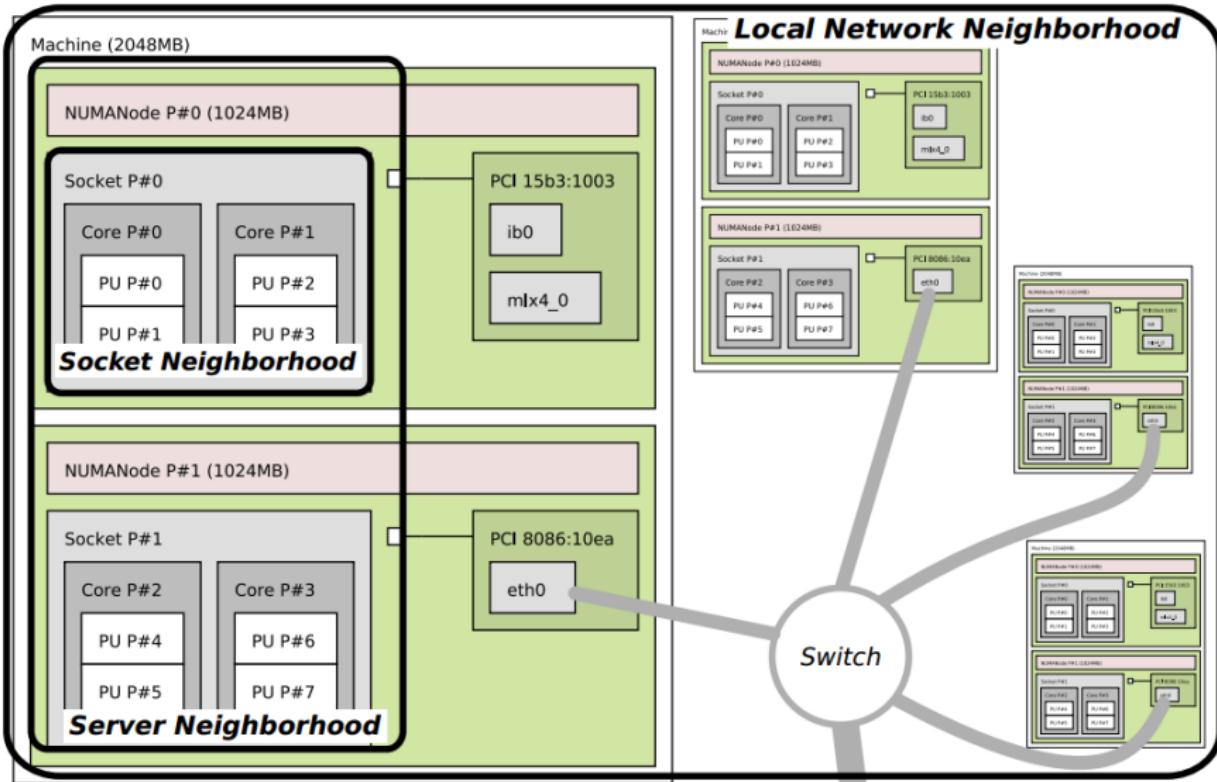


Access type	Local access	Neighbor-node access	Opposite-node access
Read	83 ns	98 ns ( $\times 1.18$ )	117 ns ( $\times 1.41$ )
Write	142 ns	177 ns ( $\times 1.25$ )	208 ns ( $\times 1.46$ )

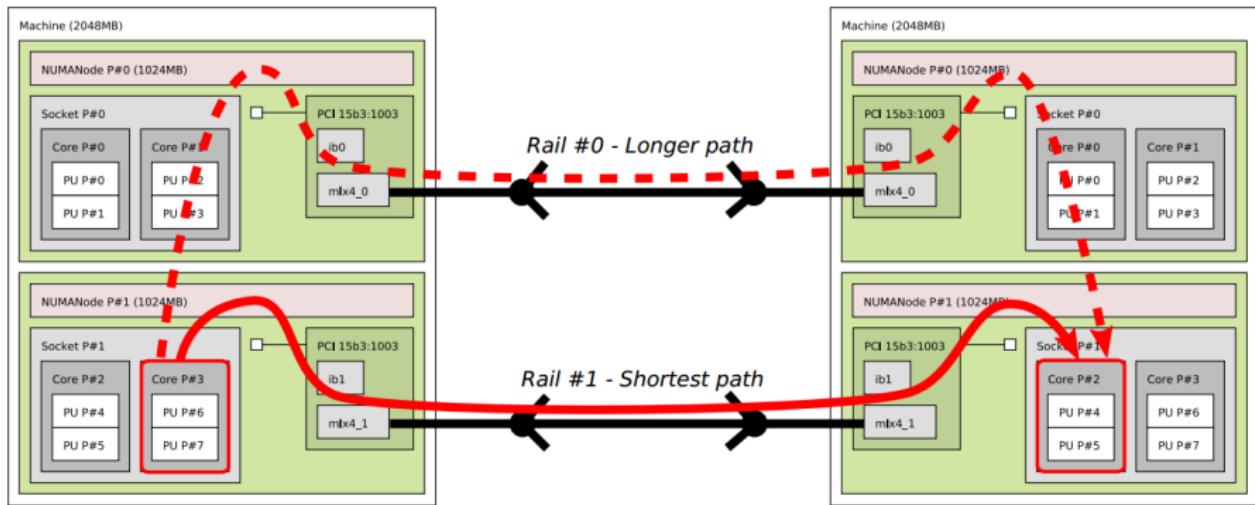
# NUIO, NUMA et GPU



# NUMA-NUIO Cluster



# NUMA-NUIO Cluster



Nécessite des runtimes (on verra si on a le temps)

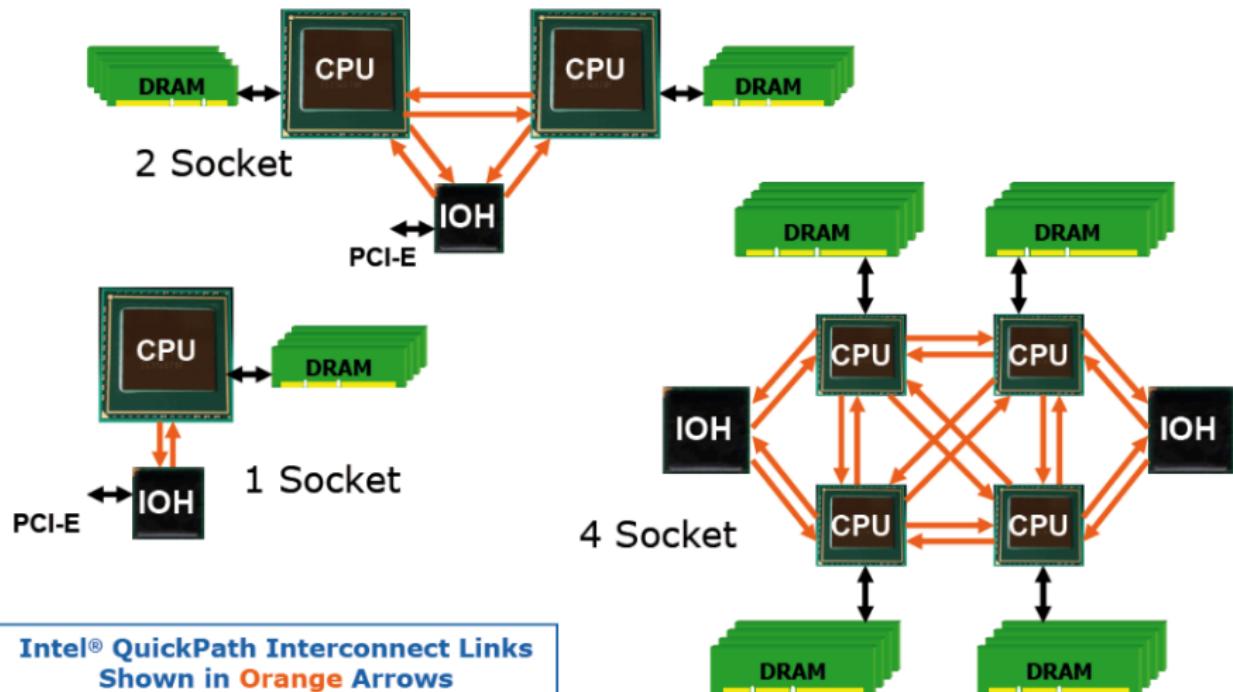
# Architecture Nehalem-EP (Intel)

- 4 cores
- Cache L3 partagé 8 Mo on-chip
- 3 niveaux de caches
- Cache L1 : 32k I-cache + 32k D-cache
- Cache L2 : 256 k par core
- Cache inclusif : cohérence de cache on-chip
- SMT
- 732 M transistors, 1 seule puce de  $263\ mm^2$
- QuickPath Interconnect
  - Point à Point
  - 2 liens par socket CPU
  - 1 pour la connexion à l'autre socket
  - 1 pour la connexion au chipset
  - QuickPath Memory controller (DDR3) intégré

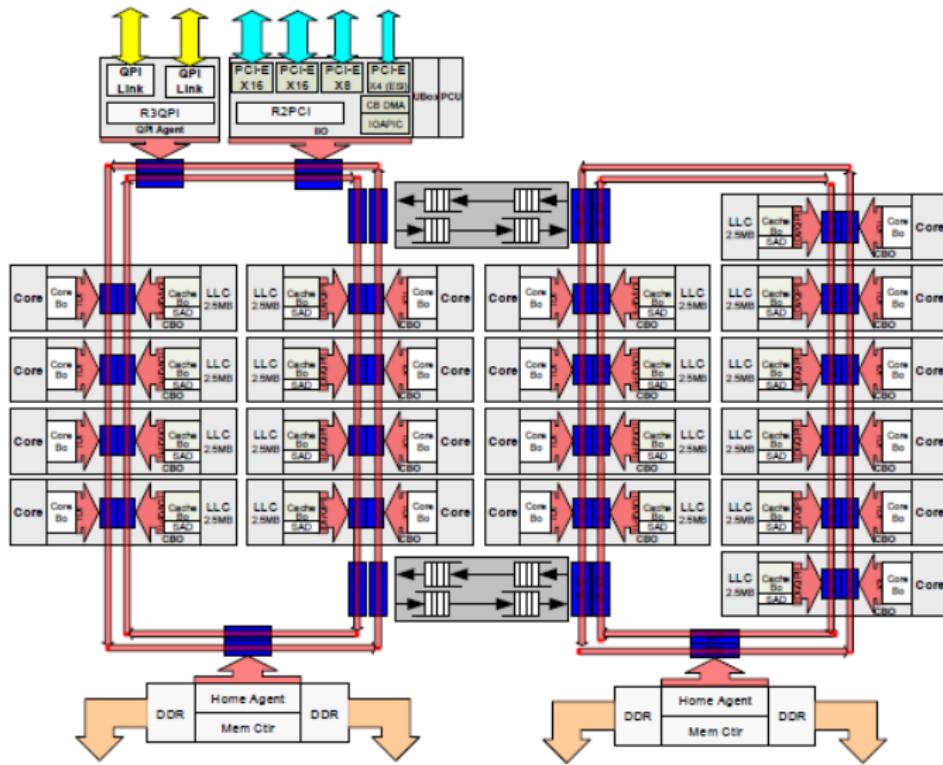


# QuickPath

## Example Platform Topologies



## Haswell

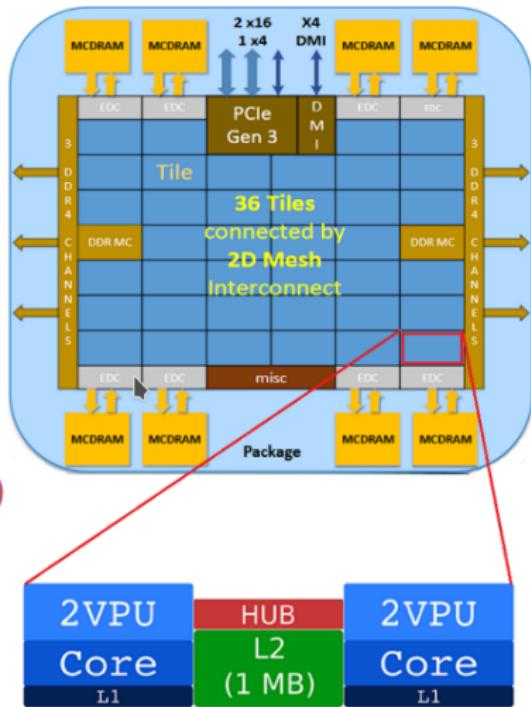


# Le processeurs MIC (Many Integrated Core) d'Intel : une réponse au GPU ?

- Des processeurs "manycores",  $\geq 50$  cores sur la même puce
- Compatibilité x86: Support des logiciels Intel
- Sortie du Xeon Phi en Juin 2012
  - 60 cores/1.053 GHz/240 threads
  - 8 GB de mémoire et 320 GB/s de bande-passante
  - 1 TFlops
  - **Mais inutilisable en pratique...**

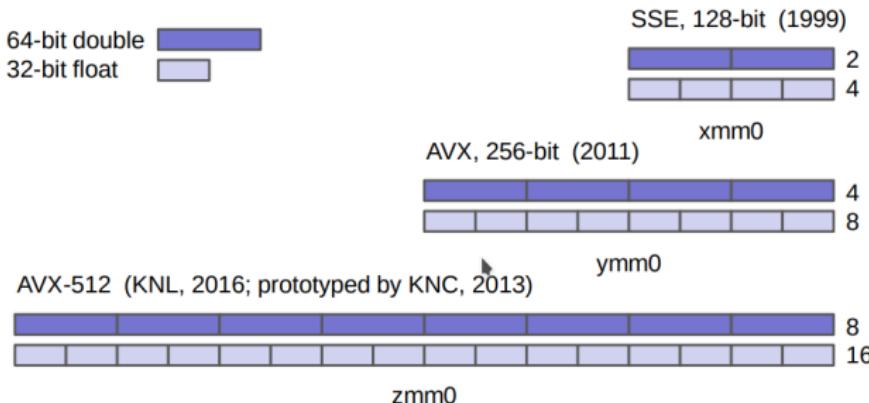
# Intel Xeon Phi KNL (Knights Landing)

- Enfin un MIC utilisable !
- Plus uniquement une carte d'extension mais un système complet
- 72 cores, 2 vector units par core



# Intel Xeon Phi KNL (Knights Landing)

- Evolution de la taille des registres de vecteurs
- Evolution des instructions de vectorisation: 1 cycle pour faire une addition et une multiplication (Fused Multiply-Add) :  $a = a \times c + b$



# Intel Xeon Phi KNL (Knights Landing)

	<u>Xeon E5</u>	<u>KNC</u>	<u>KNL</u>
Number of cores	8	61	68
SIMD width (doubles)	4	8	8 x 2
Multiply/add in 1 cycle	x 2	x 2	x 2
Clock speed (Gcycle/s)	2.7	1.01	1.4
DP Gflop/s/core	21.6	16.2	44.8
<b>DP Gflop/s/processor</b>	<b>173</b>	<b>988</b>	<b>3046</b>

# Intel Xeon Phi KNL (Knights Landing)

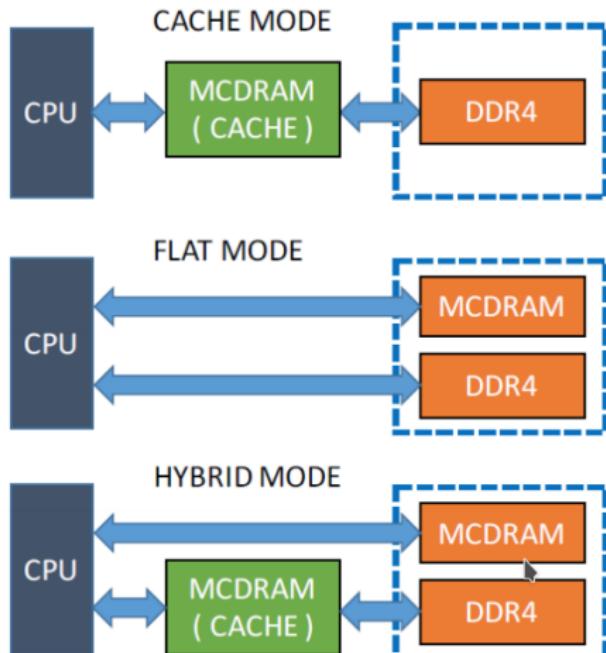
- Pas si simple à programmer car il faut exposer 2 types de parallélisme
- Parallélisme de tâches
  - OpenMP, Intel TBB, Threads
  - Comme sur les processeurs classiques
- Parallélisme de données (Vectorisation)
  - SIMD
  - Plus compliquer à exposer
  - Devient nécessaire même sur les processeurs classiques
- Pour utiliser réellement un KNL, il faut exposer les 2 types de parallismes
  - 2-4 threads par core donc 144 à 288 threads
  - 8 à 16x performance en utilisant des boucles vectorisées (4x à 8x sur les processeurs)
  - Toujours le problème de fournir suffisamment de données

# Intel Xeon Phi KNL (Knights Landing)

- Une mémoire complexe !
- 96-384GB de DRAM
  - 6 canaux de DDR4
  - Bande passante 90GB/s
- 16GB de high-speed MCDRAM
  - 8 contrôleur de DRAM intégré dans le KNL
  - Bande passante 475GB/s
- 34MB de cache L2 partagé
  - 1MB par tile (2 core)
  - Interconnexion via un mesh 2D
- 32KB de cache L1 par core
  - Accès local seulement

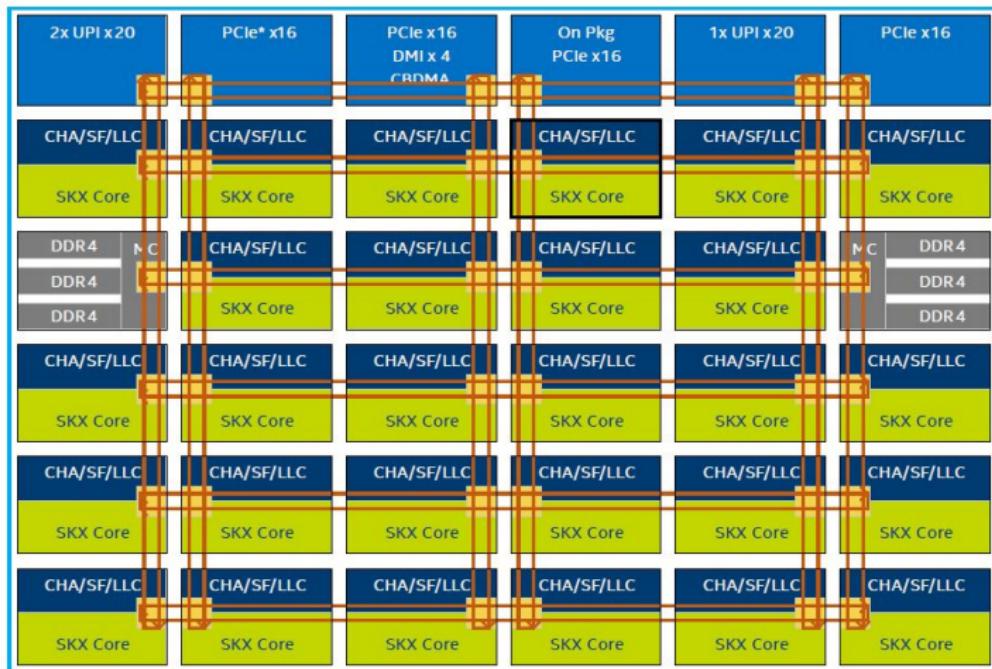
# Intel Xeon Phi KNL (Knights Landing)

- Comment utiliser la MCDRAM?
  - Plusieurs solutions
- Comme un cache
  - La MCDRAM est comme un cache L3
  - Totalement transparent pour l'utilisateur
- Plat
  - La MCDRAM et la DDR4 sont vu comme des mémoires différentes
  - Utiliser une ou modifier le code pour les 2
- Entre les deux
  - 25, 50 ou 75% de la MCDRAM est utiliser comme cache
  - le reste est accessible comme une autre mémoire



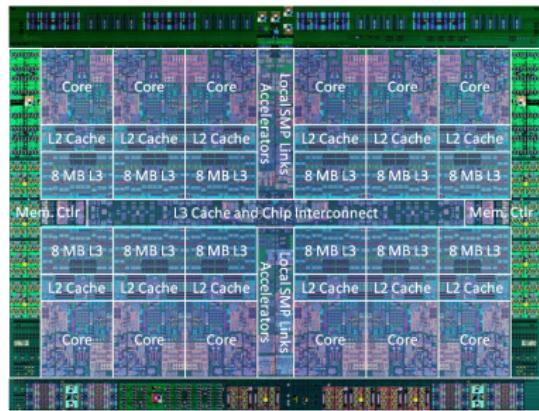
# SkyLake: KNL et Haswell

## Skylake-SP 28-core die



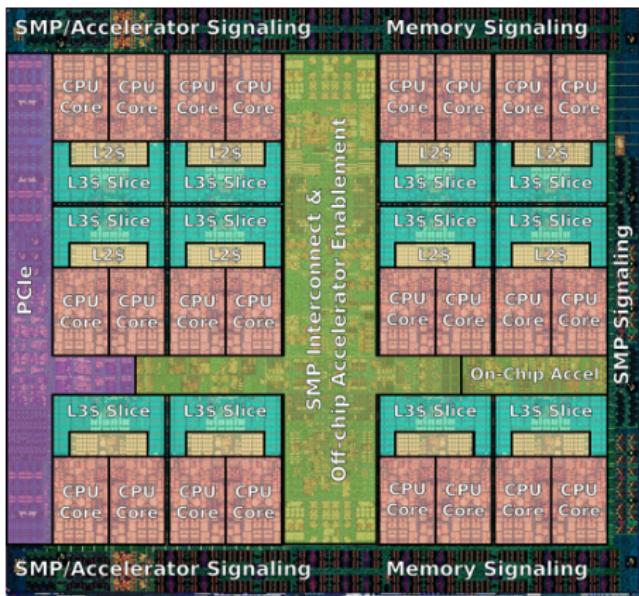
# IBM Power 8

- Jusqu'à 12 cores et 96 threads (8 par cœur)
- De 2.5 à 5Ghz
- 4 caches: 64KB, 512KB, 8Mb et 16MB
- Jusqu'à 1TB de RAM par chipset
- Jusqu'à 16 sockets (192 cœurs) et 1,536 threads !
- Compatible GPU NVidia

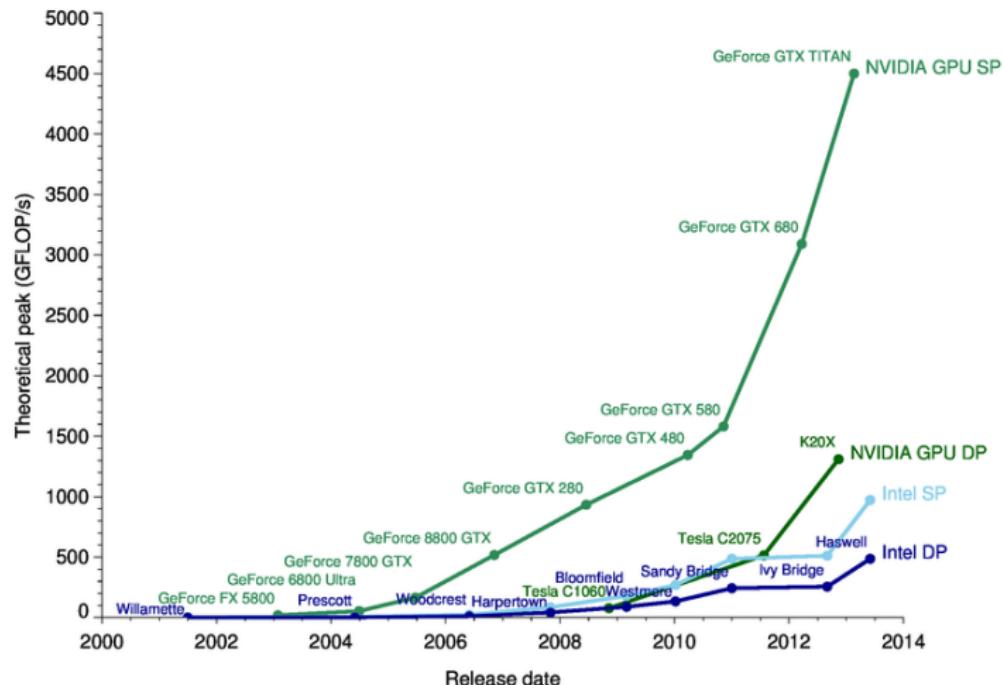


# IBM Power 9

- Jusqu'à 12-24 cores et 96-192 threads (8 par cœur)
- De 2.5 à 5Ghz
- 4 caches: 64KB, 512KB, 8Mb et 16MB
- Jusqu'à 1TB de RAM par chipset
- Jusqu'à 16 sockets (192 cœurs) et 3,072 threads !
- Compatible GPU NVidia (interface NVLink 2.0-3.0)



# Evolution des performances CPU/GPU



SP/DP (Single/Double Precision)

# Processeurs graphiques: NVidia P100 vs Intel Xeon 8180M

- 3584 CUDA cores – 8x28 cores Intel
- 5.3 TFLOPS (21.2TFLOPS halfprecision) – Xeon:  
8x2.5x28x8 (AVX512/Fréquence/Cores/Sockets)  
4.5TFLOPS
- Bande passante mémoire: 732GB/s NVidia P100 –  
150GB/s Intel Xeon (par socket) et 100GB/s entre  
sockets
- Prix : 15,000e – 100,000e (8\*28 cores)



- Présents dans tous les PC : un marché de masse.
- Adapté au massivement parallèle (milliers de threads par application).
- Jusqu'à quelques années, uniquement programmable via des APIs graphiques.
- Aujourd'hui, des modèles de programmation disponibles : CUDA , OpenACC, OpenMP

# NVidia Volta 100

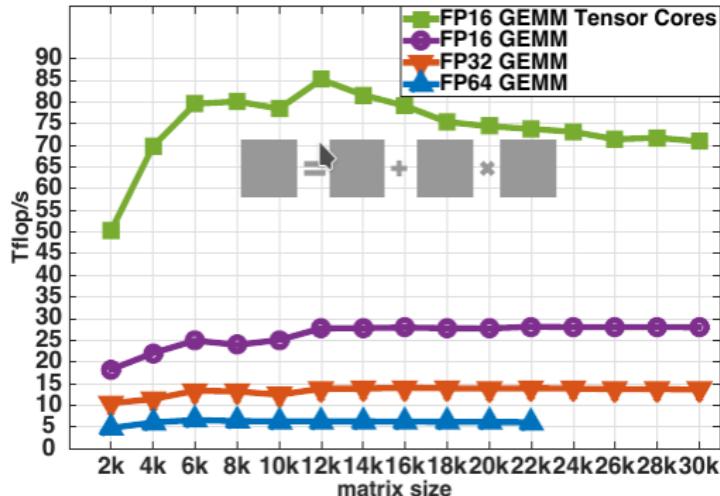
- 640 Tensor Cores (AI)
- 5,120 CUDA cores
- DP 7.8TFLOPS, SP 15.7TFLOPS, Tensor 125FLOPS
- Mémoire: 16GB (ou 32GB) HBM2
- Bande passante mémoire: 900GB/s
- NVidia NVLink: 300GB/s

## GPU PERFORMANCE COMPARISON

	P100	V100	Ratio
DL Training	10 TFLOPS	120 TFLOPS	12x
DL Inferencing	21 TFLOPS	120 TFLOPS	6x
FP64/FP32	5/10 TFLOPS	7.5/15 TFLOPS	1.5x
HBM2 Bandwidth	720 GB/s	900 GB/s	1.2x
STREAM Triad Perf	557 GB/s	855 GB/s	1.5x
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

# Tensor Cores

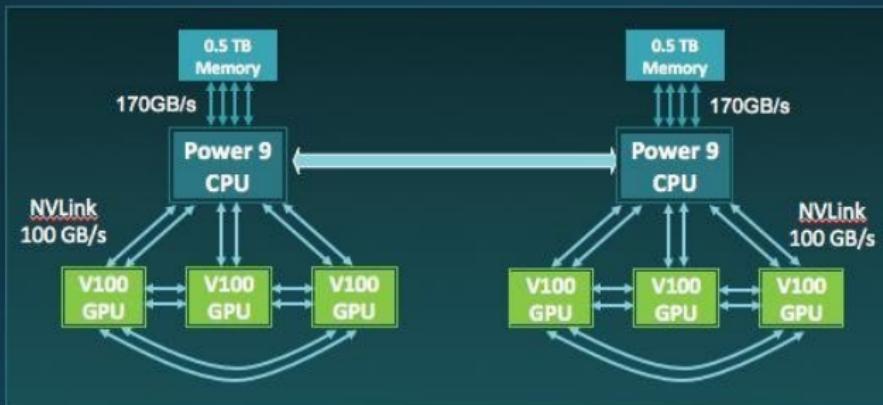
- Spécifiquement conçu pour le deep learning
- Tensor cores: Instruction FMA (Fused-Multiply-Add) sur des FP64



- Intel Nervana Neural Network Processor
- Google Tensor Processing Unit

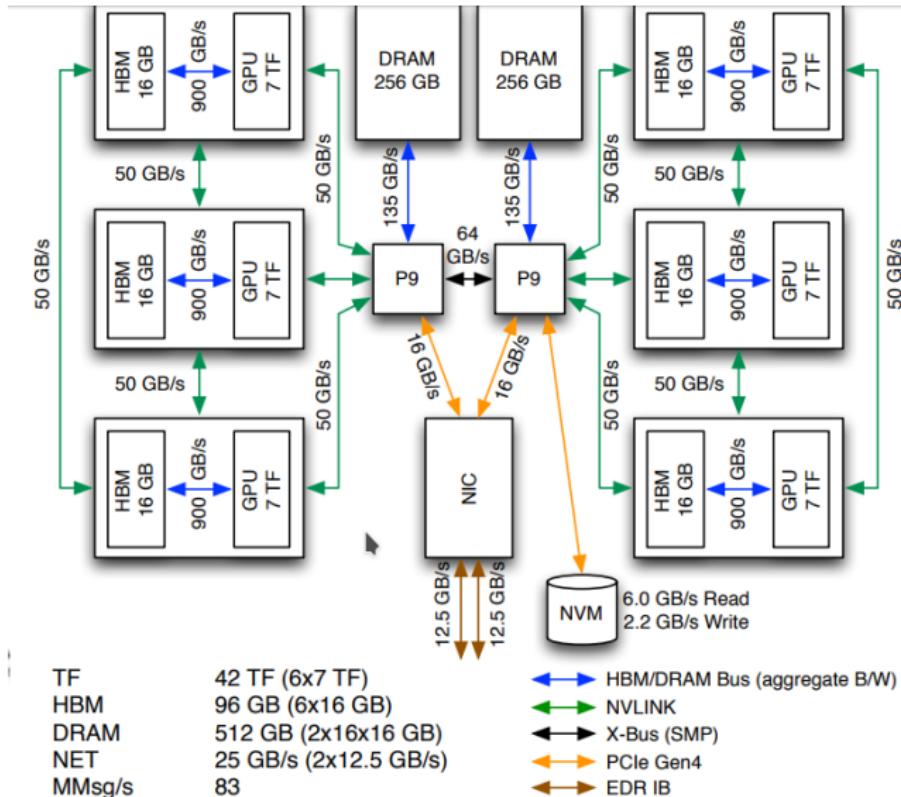
# NVLink

## Announcing New Deep Learning Server *IBM AC922 Power System*



*6x Faster CPU-GPU Data Communication  
Enables Large Models with Large Input Data*

# NVLink et Power9



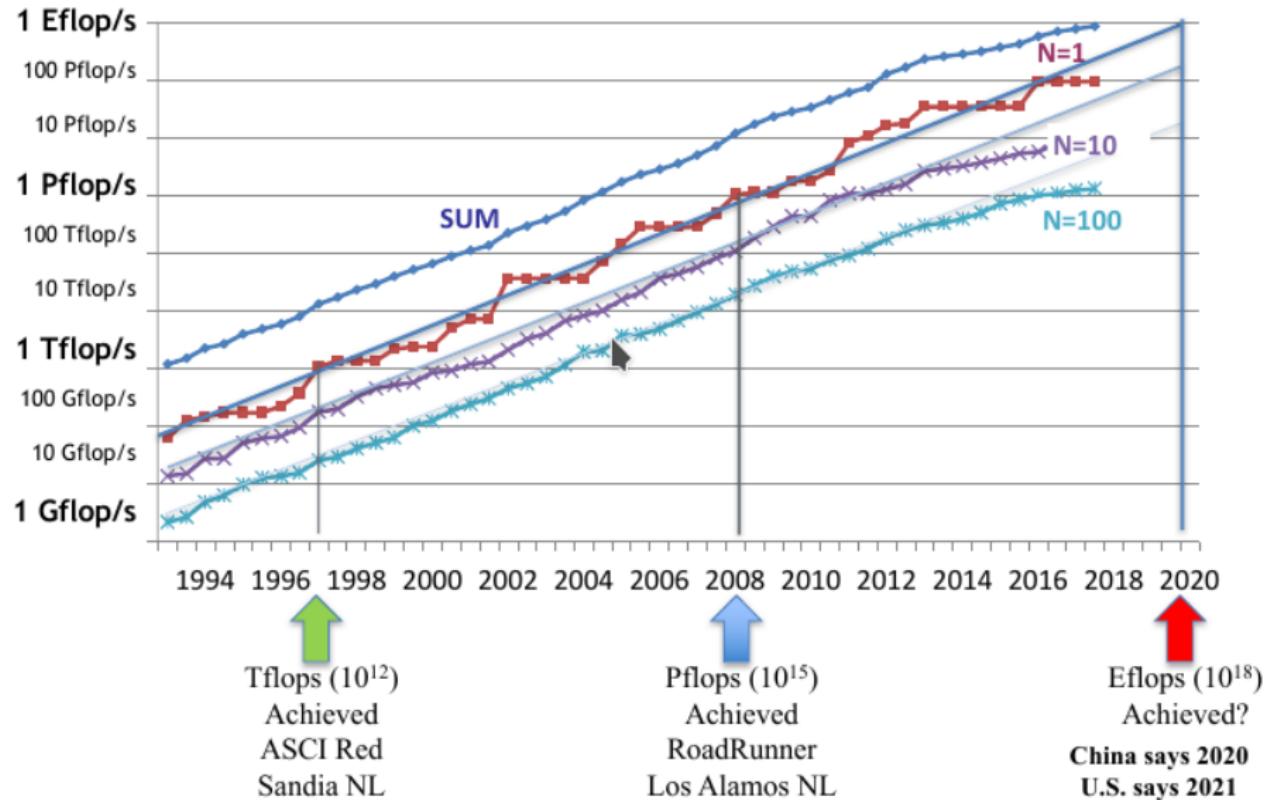
# Comparaison GPU/CPU

- A performance intrinsèque égale, les plates-formes à base de GPUs
  - occupent moins de place
  - sont moins chères
  - sont moins consommatoires d'électricité
- Mais
  - sont réservées à des applications massivement parallèles
  - nécessitent l'apprentissage de nouveaux outils
  - quelle est la garantie de pérennité des codes et donc de l'investissement en terme de portage ?

# Vers l'exascale (2018? 2020? 2021?)

Systems	2013 Titan Computer	2020	Difference Today & 2020
<b>System peak</b>	<b>27 Pflop/s</b>	<b>1 Eflop/s</b>	<b>O(100)</b>
<b>Power</b>	<b>8.3 MW (2 Gflops/W)</b>	<b>~20 MW (50 Gflops/W)</b>	<b>O(10)</b>
System memory	710 TB (38*18688)	32 - 64 PB	O(100)
Node performance	1,452 GF/s (1311+141)	1.2 or 15 TF/s	O(10)
Node memory BW	232 GB/s (52+180)	2 - 4 TB/s	O(10)
Node concurrency	16 cores CPU 2688 CUDA cores	O(1k) or 10k	O(100) - O(10)
Total Node Interconnect BW	8 GB/s	200-400 GB/s	O(100)
System size (nodes)	18,688	O(100,000) or O(1M)	O(10) - O(100)
<b>Total concurrency</b>	<b>50 M</b>	<b>O(billion)</b>	<b>O(100)</b>
<b>MTTF</b>	<b>?? unknown</b>	<b>O(&lt;1 day)</b>	<b>O(?)</b>

# Vers l'exascale



# Vers l'exascale et politique

## U.S.



- Sustained ES\*: 2022-2023
- Peak ES: 2021
- Vendors: U.S.
- Processors: U.S. (some ARM?)
- Initiatives: NSCI/ECP
- Cost: \$600M per system, plus heavy R&D investments

## EU



- PEAK ES: 2023-2024
- Pre-ES: 2021-2022
- Vendors: U.S., Europe
- Processors: Likely ARM
- Initiatives: EuroHPC
- Cost: \$300-\$350M per system, plus heavy R&D investments

## China



- Sustained ES\*: 2021-2022
- Peak ES: 2020
- Vendors: Chinese (multiple sites)
- Processors: Chinese (plus U.S.?)
- 13<sup>th</sup> 5-Year Plan
- Cost: \$350-\$500M per system, plus heavy R&D

## Japan



- Sustained ES\*: ~2022
- Peak ES: Likely as a AI/ML/DL system
- Vendors: Japanese
- Processors: Japanese
- Cost: \$800M-\$1B, this includes both 1 system and the R&D costs, will also do many smaller size systems