Analyse de performance

SAE Semestre 5

I. Sommaire

II. Introduction

A. Présentation

Cette expérience a pour objectif d'analyser la performance et la scalabilité de trois programmes de calcul distribué :

1. L'estimation du nombre π par la méthode de Monte Carlo.
2. Le calcul de nombres premiers utilisant MPI.
3. Le calcul d'intégrale par la méthode de Simpson

L'objectif est de déterminer dans quelle mesure ces programmes conservent des performances acceptables lorsque l'on augmente les ressources de calcul disponibles, notamment en termes de nombre de threads ou de taille des données traitées.

Pour la méthode de Monte Carlo, nous réaliserons d'abord un test d'erreur relative avant. Comme ce programme utilise le hasard pour ses calculs, cette étape de vérification est nécessaire pour s'assurer que le résultat obtenu est correct et suffisamment précis avant de commencer à mesurer la vitesse d'exécution.

Cette analyse s'appuiera sur les normes ISO 25010 et ISO 25022, qui définissent les critères de qualité logicielle et les métriques permettant d'évaluer la performance, notamment la scalabilité, le temps d'exécution et l'utilisation des ressources.

B. Environnement

Pour réaliser ces expériences, nous utiliserons un cluster de Raspberry pi composé de :

- 1 Raspberry Pi 4 :
  - Processeur : 4 cœurs
  - Fréquence : 1.8 GHz
- 4 Raspberry Pi 0 :
  - Processeur : 1 cœur
  - Fréquence : 1.0 GHz

C. Scalabilité

1. Définition

La scalabilité est la capacité d'un programme informatique à continuer de fonctionner efficacement lorsqu'on augmente les ressources disponibles ou la quantité de travail à traiter.

$$S_p = \frac{T_1}{T_p}$$

Pour mesurer cette évolution des performances, on utilisera un indicateur appelé speedup. Le speedup permet de comparer le temps d'exécution d'un programme avec un seul processeur à celui obtenu avec plusieurs processeurs.

2. Scalabilité Forte (Strong Scaling)

La Scalabilité Forte a pour objectif de mesurer si un programme s'exécute plus vite quand on augmente le nombre de processeurs sans changer la taille du problème.

Le speedup doit suivre une fonction linéaire : si l'on double le nombre de processeurs, on devrait théoriquement diviser le temps d'exécution par deux. Plus la courbe du speedup se rapproche d'une droite linéaire, plus le programme présente une bonne scalabilité forte.

3. Scalabilité Faible (Weak Scaling)

La Scalabilité Faible a pour objectif de mesurer si un programme garde le même temps d'exécution quand on augmente le nombre de processeurs tout en augmentant la taille totale du problème.

Un programme avec une faible scalabilité doit maintenir un speedup proche d'une fonction constante égale à 1 car l'augmentation de ressources doit compenser l'augmentation de la charge de travail, ce qui permet de conserver un temps d'exécution stable.

III. Analyse de performance de Monte Carlo

A. Analyse de la scalabilité forte

1. Objectif

Mesurer si le programme d'estimation du nombre π par la méthode de Monte Carlo s'exécute plus vite quand on augmente le nombre de processeurs sans changer la taille du problème.

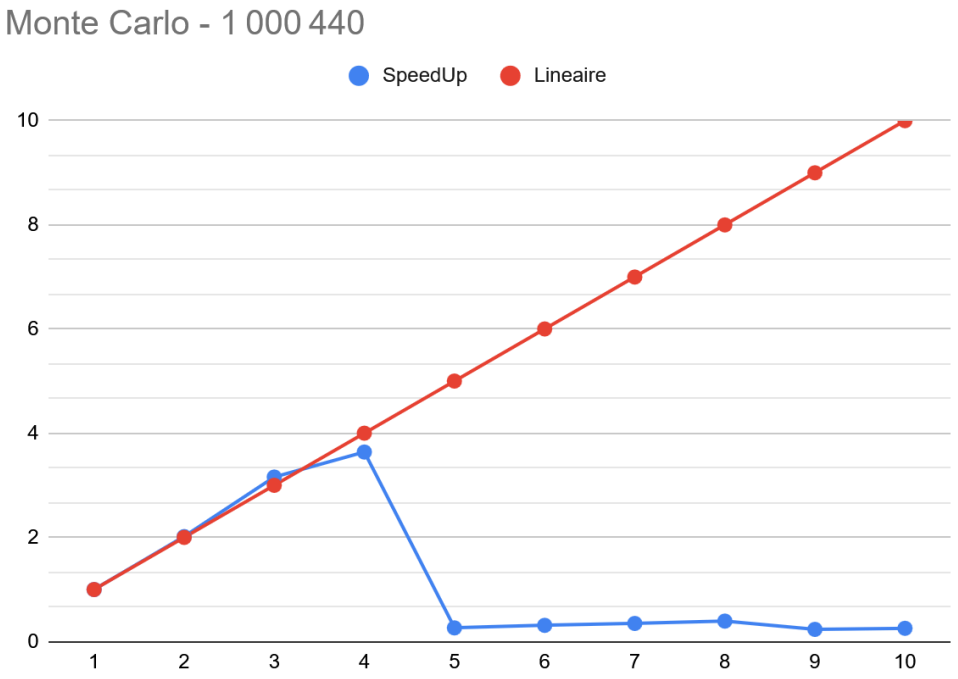
Pour réaliser cette expérience, on mesurera le temps d'exécution de 1 000 440, 10 001 880 et 100 001 160 exécutions en utilisant 1 à 10 threads.

On effectuera à chaque fois 5 expériences similaires dont on gardera la valeur médiane de ces expériences.

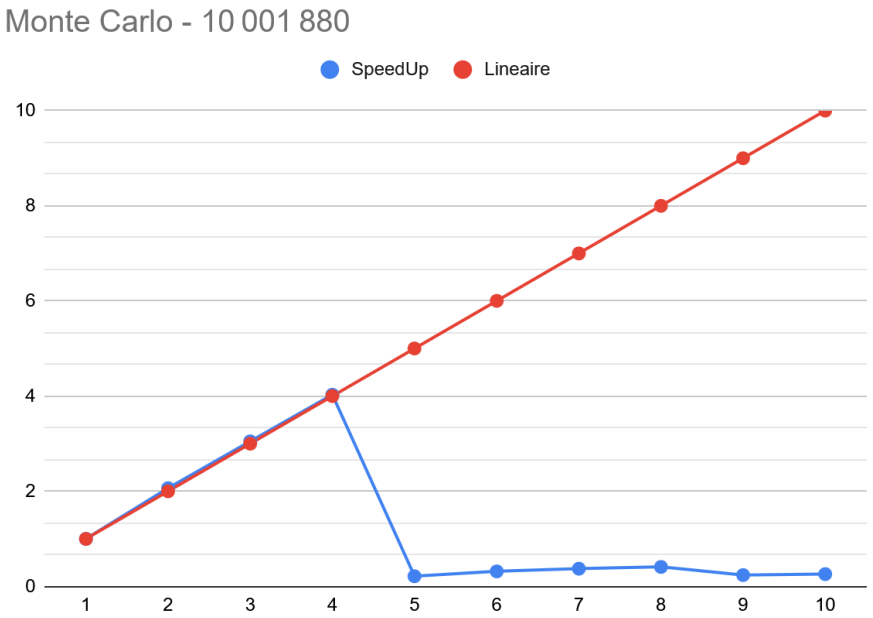
2. Résultats

Après avoir réalisé nos expériences, nous obtenons les tableaux et les graphiques suivants :

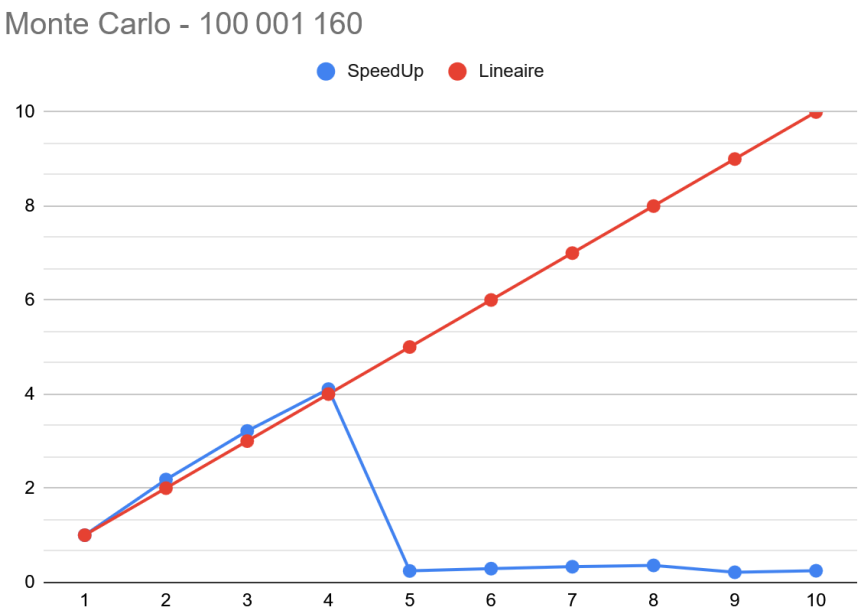
nb tot	1 000 440	1 000 440	1 000 440	1 000 440	1 000 440	1 000 440	1 000 440	1 000 440	1 000 440	1 000 440
nb tot par wk	1000440	500220	333480	250110	200088	166740	142920	125055	111160	100044
nb processus p	1	2	3	4	5	6	7	8	9	10
Sp	1	2,017256786	3,158185466	3,638131605	0,2644185883	0,313513822	0,3489147916	0,3921328262	0,2335296849	0,2526174158
Tp	133674116	66265295	42326240	36742518	505539784	426373916	383113927	340889890	572407384	529156375
Res1	190827687	149862433	19285088	213633862	924043985	842659157	777218576	745920725	139870012	1362207126
Res2	148902055	85660377	42326240	100820815	500141339	426373916	385254497	351255230	573074618	529564020
Res3	133674116	66265295	10930685	11043518	505539784	425630905	383113927	338744208	575092931	529156375
Res4	133357230	65961224	46364347	36742518	522605538	421425615	364950925	340889890	572407384	525752688
Res5	133084714	65896354	46626456	36418632	482844920	439048566	379765183	336931597	569383266	517828287



10 001 880	10 001 880	10 001 880	10 001 880	10 001 880	10 001 880	10 001 880	10 001 880	10 001 880	10 001 880	10 001 880
nb tot par wk	10001880	5000940	3333960	2500470	2000376	1666980	1428840	1250235	1111320	1000188
nb processus p	1	2	3	4	5	6	7	8	9	10
Sp	1	2,063251795	3,047694389	4,027675903	0,2138296384	0,3162849371	0,3718985467	0,4094000365	0,2367932765	0,2588519445
Tp	1364287727	661231814	447645844	338728279	6380255502	4313476764	3668440598	3332407439	5761513787	5270533044
Res1	1283976446	638698821	651565819	663567568	6810414395	4722808702	4339696884	3721134544	6622715521	6548165713
Res2	1319824935	656062294	500220331	472313299	6380255502	4305044832	3663360483	3332407439	5746250146	5299782694
Res3	1364588039	702630304	442995687	337775140	6371715605	4313476764	3663919440	3294164011	5761513787	5251621365
Res4	1364287727	661465368	447645844	336443542	6410473065	4315940145	3683008542	3320458430	5764598533	5270533044
Res5	1364380281	661231814	442372898	338728279	6376040189	4310785234	3668440598	3416356848	5741002660	5268287418



nb tot	100 001 160	100 001 160	100 001 160	100 001 160	100 001 160	100 001 160	100 001 160	100 001 160	100 001 160	100 001 160
nb tot par wk	100001160	50000580	33333720	25000290	20000232	16666860	14285880	12500145	11111240	10000116
nb processus p	1	2	3	4	5	6	7	8	9	10
Sp	1	2,181339529	3,213906994	4,107093142	0,2399476072	0,2879919841	0,3297077941	0,3560494999	0,2109129302	0,2431760671
Tp	13065611985	5989719534	4065336057	3181230991	54451936977	45367971012	39627852957	36696054868	61947894686	53729020864
Res1	13055533011	5989719534	4002115827	3184079604	53903583144	45032443561	39627852957	36696054868	62748390898	54628203089
Res2	13022708842	5909195264	4065336057	3228199196	54838870428	45655284351	39687005544	36627013682	61947894686	54370250773
Res3	13302250968	5930161557	4051070192	3170599592	54335916175	45367971012	39831209613	36898484688	61786409221	53655371827
Res4	13146448222	6028552261	4090764723	3166316133	54451936977	45765409439	39528702818	37027127476	62429237726	53729020864
Res5	13065611985	6039851181	4079603181	3181230991	55344976315	44969321092	39125592463	36653463836	61756657979	53646853585



Sur ces graphiques :

- La courbe Linéaire désigne la courbe attendue pour un programme avec une scalabilité forte optimale.
- La courbe SpeedUp désigne la courbe de scalabilité forte obtenue après l'expérience menée sur le code Monte Carlo.

3. Analyse des résultats

Observation : Peu importe la taille du problème (10 puissance 6, 10 puissance 7 ou 10 puissance 8), la courbe a toujours la même forme.
De 1 à 4 processus : Le Speedup monte de façon linéaire et atteint 4 presque partout, ce qui est le résultat idéal.

À partir de 5 processus : Les courbes chutent brutalement. Le Speedup tombe en dessous de 0,3, ce qui signifie que le calcul devient plus lent qu'avec un seul processeur.

À partir de 9 processus : Le speedUp tombe encore plus bas, en dessous de 0,2.

B. Analyse de la scalabilité faible

1. Objectif

Mesurer si le programme d'estimation du nombre π par la méthode de Monte Carlo garde le même temps d'exécution quand on augmente le nombre de threads tout en augmentant la taille totale du problème proportionnellement.

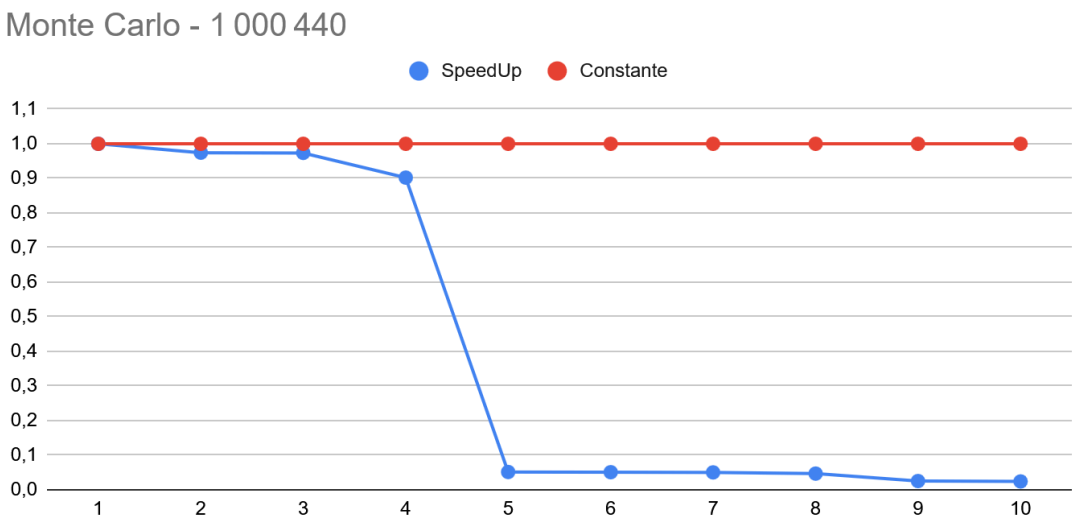
Pour réaliser cette expérience, on mesurera le temps d'exécution pour 1 000 440, 10 001 880 et 100 001 160 exécutions par thread. On effectuera cette expérience avec 1 à 10 threads.

On effectuera à chaque fois 5 expériences similaires dont on gardera la valeur médiane de ces expériences.

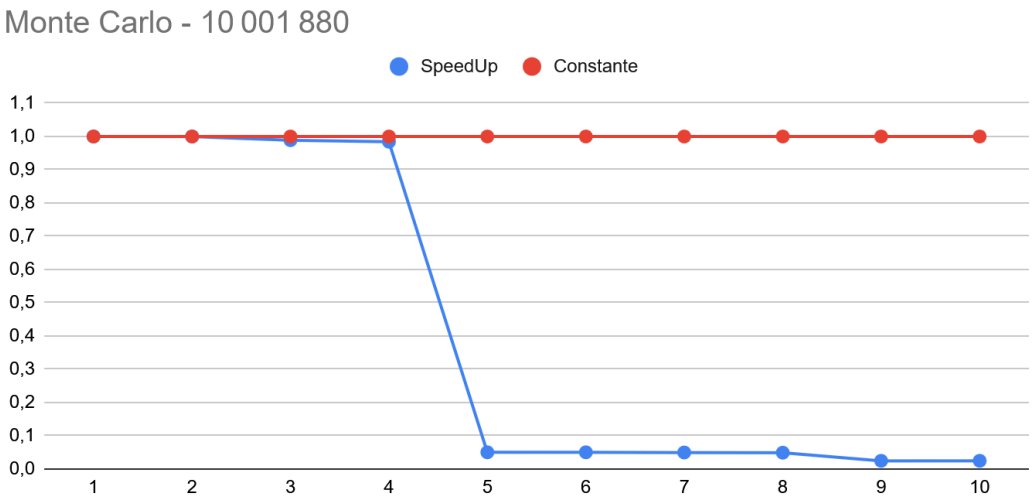
2. Résultats

Après avoir réalisé nos expériences, nous obtenons les tableaux et les graphiques suivants :

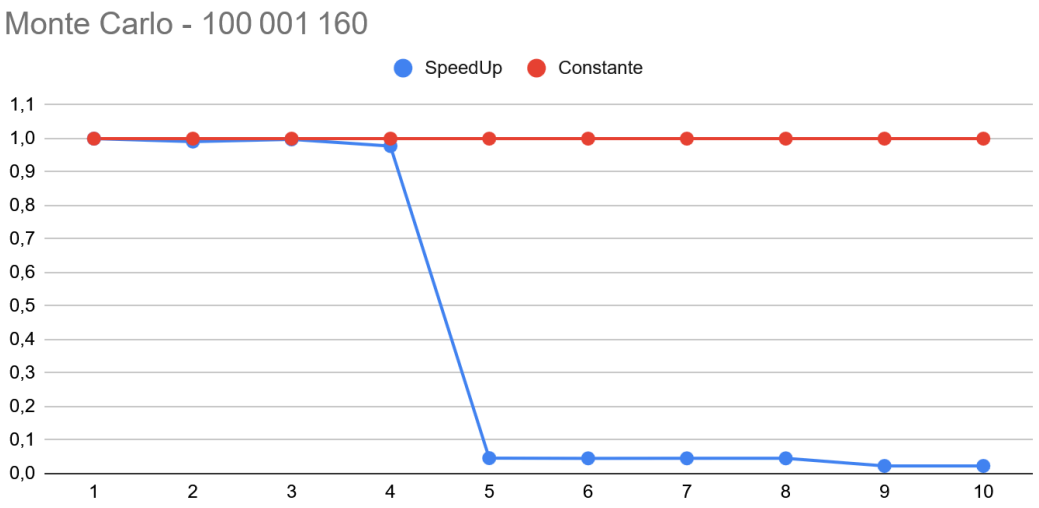
nb tot	1 000 440	2 000 880	3 001 320	4 001 760	5 002 200	6 002 640	7 003 080	8 003 520	9 003 960	10 004 400
nb processus p	1	2	3	4	5	6	7	8	9	10
constante k	1	1	1	1	1	1	1	1	1	1
Sp	1	0,9739965526	0,9733203044	0,9022453882	0,05141166325	0,05097577516	0,0502741525	0,04705144038	0,02552866944	0,02447517656
Tp	133404555	136966147	137061309	147858395	2594830561	2617018664	2653541599	2835291628	5225675991	5450606440
Res1	173404335	198944341	317007056	355729341	3008610315	3089989250	3056730149	3060377370	6062395916	6051671614
Res2	145237450	157595613	273959786	276173172	2594830561	2696933961	2653541599	2656400001	5224012939	5299618116
Res3	133404555	135691125	137061309	138091651	2566606189	2610053621	2654322718	2679149071	5232014658	5290583652
Res4	133265816	136966147	136011172	137325067	2594604865	2617018664	2629556301	2865966351	5223626356	5455217508
Res5	133138485	135875493	136814957	147858395	2596184952	2609716739	2652696554	2835291628	5225675991	5450606440



nb tot	10 001 880	20 003 760	30 005 640	40 007 520	50 009 400	60 011 280	70 013 160	80 015 040	90 016 920	100 018 800
nb processus p	1	2	3	4	5	6	7	8	9	10
constante k	1	1	1	1	1	1	1	1	1	1
Sp	1	0,9996379423	0,9887214056	0,9841323736	0,05052551905	0,05042224925	0,04952624242	0,04906778049	0,02463635287	0,02471783747
Tp	1364150167	1364644247	1379711372	1386144998	26999231136	27054528255	27543986792	27801342414	55371433181	55188896233
Res1	1220907136	1278188882	1334278619	1467461842	27670955933	27495839885	27862687946	28155004925	56007628543	55936285493
Res2	1354215810	1336494735	1428143756	1489588327	26999231136	27022576597	27517868775	27734088611	56099694289	55188896233
Res3	1364274183	1368784777	1379711372	1385051305	26973429645	27018522435	27731053090	27782936424	55137983571	55098557362
Res4	1364150167	1364644247	1388691942	1386144998	27256694335	27054528255	27521106351	27943278954	55114669874	55221432752
Res5	1364372407	1366255880	1366658421	1379465980	26983953193	27222961696	27543986792	27801342414	55371433181	55035583779



nb tot	100 001 160	200 002 320	300 003 480	400 004 640	500 005 800	600 006 960	700 008 120	800 009 280	900 010 440	1 000 011 600
nb processus p	1	2	3	4	5	6	7	8	9	10
constante k	1	1	1	1	1	1	1	1	1	1
Sp	1	0,9909671658	0,9976358542	0,9777165814	0,04634722327	0,04562711428	0,04593052561	0,04587739362	0,02312946834	0,02315745812
Tp	13287981911	13409104126	13319471083	13590832112	286705027236	291229943458	289306114697	289641168844	574504425137	573810037481
Res1	13287981911	13282088912	13319471083	13529208258	285607674706	291229943458	288883062739	289641168844	576504044685	575260188065
Res2	13382077289	13485841661	13093655178	13771678187	287840809408	289407635197	291630751098	289765287916	574504425137	573810037481
Res3	13094933374	13336344118	13391141205	13487728422	287536617493	291625382582	288986288627	286757339765	573974486961	576542711138
Res4	13310881842	13409104126	13321492802	13590832112	285986354946	292109611439	289306114697	290567448567	576599181455	573289595636
Res5	13279836624	13440519281	13154200869	13699625727	286705027236	289411990679	290875365128	287346645332	573803239044	572883072388



Sur ces graphiques :

- La courbe Constante désigne la courbe attendue pour un programme avec une scalabilité faible optimale.
- La courbe SpeedUp désigne la courbe de scalabilité faible obtenue après l'expérience menée sur le code Monte Carlo.

Observation : De 1 à 4 processus, le Speedup reste presque parfaitement stable à 1 sur les 3 expériences, ce qui valide la scalabilité faible sur le rpi4. Cependant, dès le passage à 5 processus, le speedup baisse aux alentours de 0,05. Puis 0,25 à partir du 9e.

C. Analyse de l'erreur relative avant

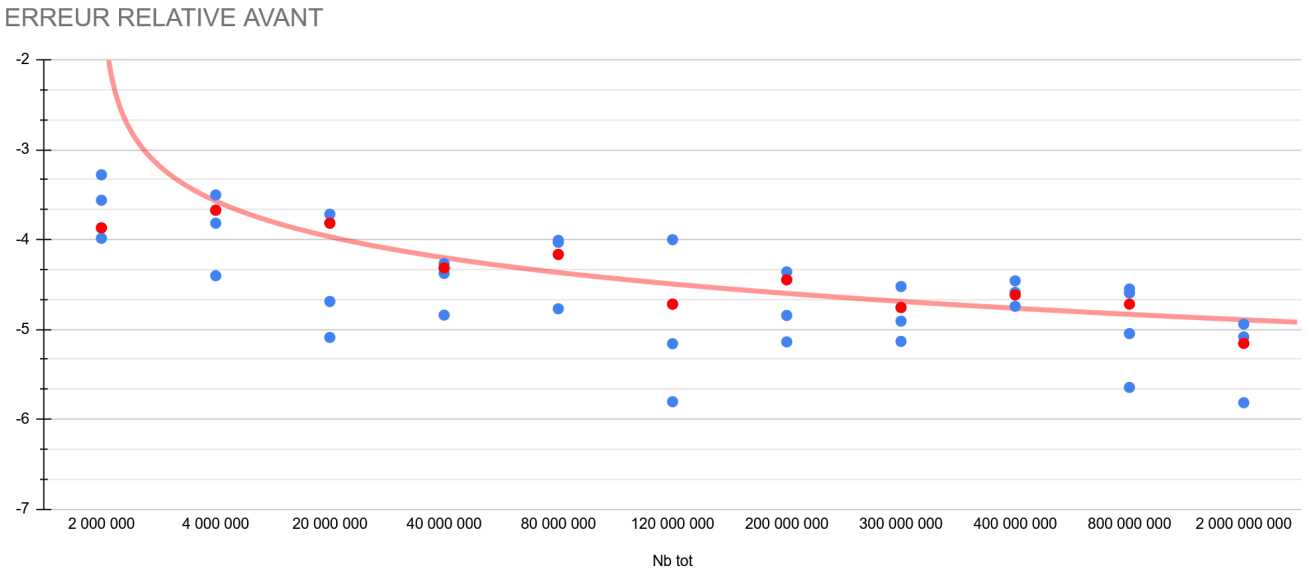
1. Objectif

Le but de cette expérience est de vérifier la fiabilité de l'algorithme de Monte Carlo. L'objectif est de déterminer à partir de quel nombre d'itérations (tirages) l'estimation de π devient suffisamment précise (erreur négligeable).

2. Résultats

Après avoir réalisé nos expériences, nous obtenons le tableau et le graphique suivant :

Nb tot	2 000 000	4 000 000	20 000 000	40 000 000	80 000 000	120 000 000	200 000 000	300 000 000	400 000 000	800 000 000	2 000 000 000
Nb worker	4	4	4	4	4	4	4	4	4	4	4
Log10(Erreur) 1	-3,279493932	-3,503425565	-3,818400669	-4,841565744	-4,772025992	-4,719334681	-4,361338186	-4,523469138	-4,615044405	-4,551540535	-5,817321464
Log10(Erreur) 2	-3,562981606	-3,819226976	-5,089819991	-4,378047169	-4,009875133	-5,805796313	-5,138712255	-4,756368276	-4,587345691	-5,047341285	-5,155509424
Log10(Erreur) 3	-3,987481829	-4,402516621	-4,689977516	-4,316297178	-4,031984833	-4,001864973	-4,447132204	-5,134662564	-4,460655078	-5,645805273	-4,940818297
Log10(Erreur) 4	-3,871165311	-3,67344981	-3,719471964	-4,268355466	-4,166743385	-5,160284924	-4,845615912	-4,908315277	-4,743462735	-4,595929142	-5,084682434
Log10(Erreur) 5	-4,149520171	-3,432188091	-3,661425326	-4,219205246	-5,1053962	-4,009518194	-4,024413296	-4,678134364	-5,455504178	-4,719914575	-6,060286206
Erreur1	5,25E-04	3,14E-04	1,52E-04	1,44E-05	1,69E-05	1,91E-05	4,35E-05	3,00E-05	2,43E-05	2,81E-05	1,52E-06
Erreur2	2,74E-04	1,52E-04	8,13E-06	4,19E-05	9,78E-05	1,56E-06	7,27E-06	1,75E-05	2,59E-05	8,97E-06	6,99E-06
Erreur3	1,03E-04	3,96E-05	2,04E-05	4,83E-05	9,29E-05	9,96E-05	3,57E-05	7,33E-06	3,46E-05	2,26E-06	1,15E-05
Erreur4	1,35E-04	2,12E-04	1,91E-04	5,39E-05	6,81E-05	6,91E-06	1,43E-05	1,24E-05	1,81E-05	2,54E-05	8,23E-06
Erreur5	7,09E-05	3,70E-04	2,18E-04	6,04E-05	7,85E-06	9,78E-05	9,45E-05	2,10E-05	3,50E-06	1,91E-05	8,70E-07
Mediane	1,35E-04	2,12E-04	1,52E-04	4,83E-05	6,81E-05	1,91E-05	3,57E-05	1,75E-05	2,43E-05	1,91E-05	6,99E-06
Log10 Mediane	-3,871165311	-3,67344981	-3,818400669	-4,316297178	-4,166743385	-4,719334681	-4,447132204	-4,756368276	-4,615044405	-4,719914575	-5,155509424



3. Analyse des Résultats

Observation : La courbe décroît de manière constante. Pour un faible nombre de tirages (2 000 0000), l'erreur est encore visible, mais elle devient négligeable en augmentant le

nombre de tirages jusqu'à atteindre une erreur de moins de 10 puissances -5 lorsque l'on atteint 2 000 000 000 itérations.

Interprétation : Cette observation valide la fiabilité de l'algorithme de Monte Carlo, car plus la charge de calcul est élevée, plus l'estimation de π est précise.

IV. Analyse de performance de MPI

A. Analyse de la scalabilité forte

1. Objectif

Mesurer si le programme de calcul de nombres premiers utilisant MPI s'exécute plus vite quand on augmente le nombre de processeurs sans changer la taille du problème.

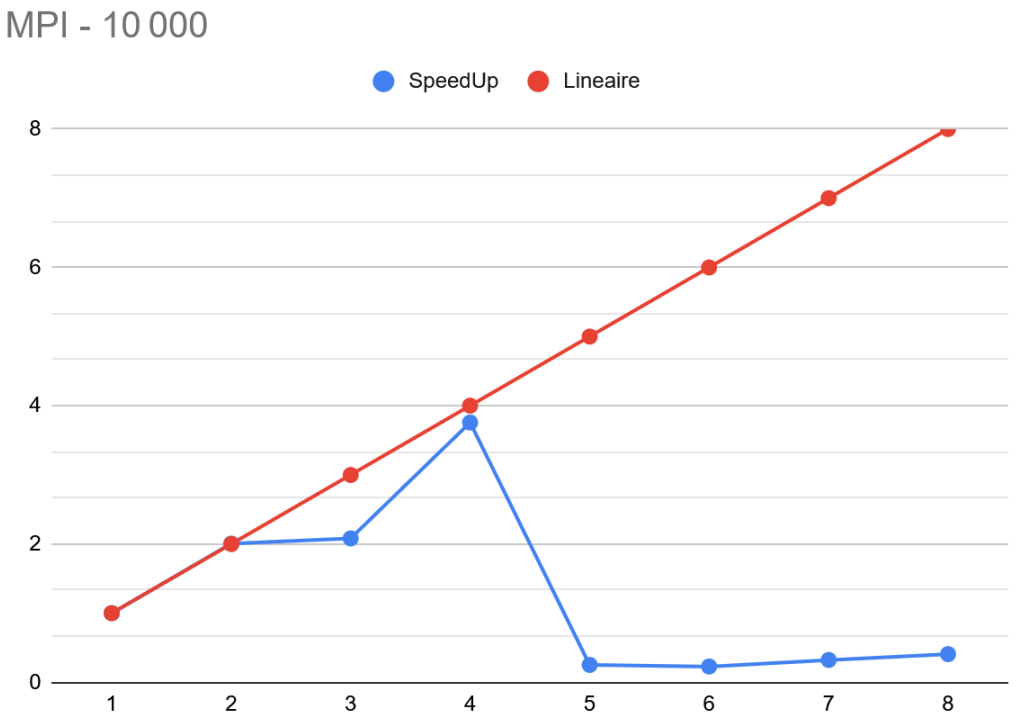
Pour réaliser cette expérience, on mesurera le temps d'exécution pour la recherche de nombre premier jusqu'à 10 000 et 40 000 en utilisant 1 à 8 cœurs.

On effectuera à chaque fois 5 expériences similaires dont on gardera la valeur médiane de ces expériences.

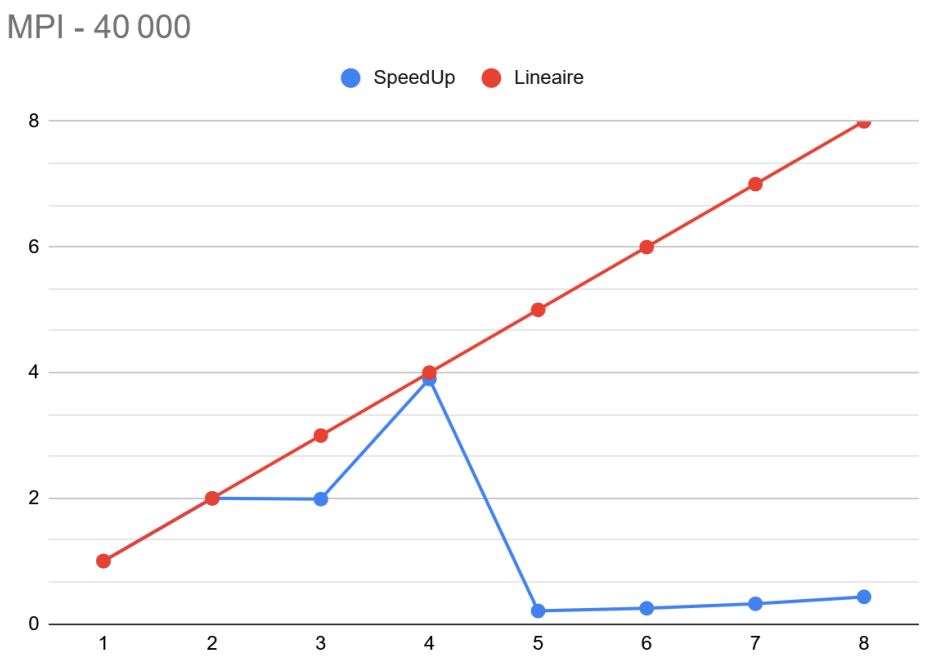
2. Résultats

Après avoir réalisé nos expériences, nous obtenons les tableaux et les graphiques suivants :

nb tot	10 000	10 000	10 000	10 000	10 000	10 000	10 000	10 000
nb processus	1	2	3	4	5	6	7	8
Sp	1	2,006444295	2,082446193	3,758090613	0,254013336	0,2286569001	0,3227302905	0,408320997
Tp	1114390518	555405660	535135324	296531040	4387133902	4873636079	3453008753	2729202089
Res1	1114390518	592978300	571073864	344896621	4095399032	4733569741	3143387343	2555401989
Res2	1123476435	550144439	549473522	280592754	4204981261	4873636079	4026015169	2907305827
Res3	1057661552	614125587	535135324	316003670	4904732380	5813799309	2817822782	2991015119
Res4	1125759661	543054089	527524762	296531040	5922143093	5035047156	3891101704	2729202089
Res5	1094882584	555405660	530787833	280181363	4387133902	4839323986	3453008753	2577400689



nb tot	40 000	40 000	40 000	40 000	40 000	40 000	40 000	40 000
nb processus	1	2	3	4	5	6	7	8
Sp	1	1,998956933	1,987741435	3,900339772	0,2082625123	0,2498407337	0,3205341985	0,4302938555
Tp	15102059687	7554970012	7597597665	3871985665	72514537144	60446747266	47115283667	35097084220
Res1	15282386310	8078467393	7368931844	3819767756	61676814029	60446747266	39182542621	40012062256
Res2	15526060072	7487945787	7597597665	4087263499	72514537144	58997967443	52769629295	45657069686
Res3	14843614412	7798031145	8302748916	3871985665	77396248200	63956896122	51254413273	34410470787
Res4	15102059687	7554970012	7440115019	3972503334	53900887778	74265906884	44943189160	33207618778
Res5	14776626998	7498982920	7759007093	3757945531	74422364373	57581615168	47115283667	35097084220



Sur ces graphiques :

- La courbe Linéaire désigne la courbe attendue pour un programme avec une scalabilité forte optimale.
- La courbe SpeedUp désigne la courbe de scalabilité forte obtenue après l'expérience menée sur le code MPI.

3. Analyse des Résultats

Observation : Comme pour Monte Carlo, les performances sont excellentes jusqu'à 4 processus, avec un Speedup qui monte jusqu'à 3,9 dans le meilleur des cas.
On peut voir une stagnation entre 2 et 3 processus : le Speedup ne progresse pas et reste bloqué aux alentours de 2, alors qu'il devrait théoriquement monter.

Cependant, dès l'ajout du 5e processus, les performances chutent fortement pour atteindre un Speedup d'environ 0,2. Que l'on cherche les nombres premiers jusqu'à 10 000 ou 40 000, le comportement reste identique : une montée optimale suivie d'une chute brutale.

B. Analyse de la scalabilité faible

1. Objectif

Mesurer si le programme de calcul de nombres premiers utilisant MPI garde le même temps d'exécution quand on augmente le nombre de processeurs tout en augmentant la taille totale du problème proportionnellement.

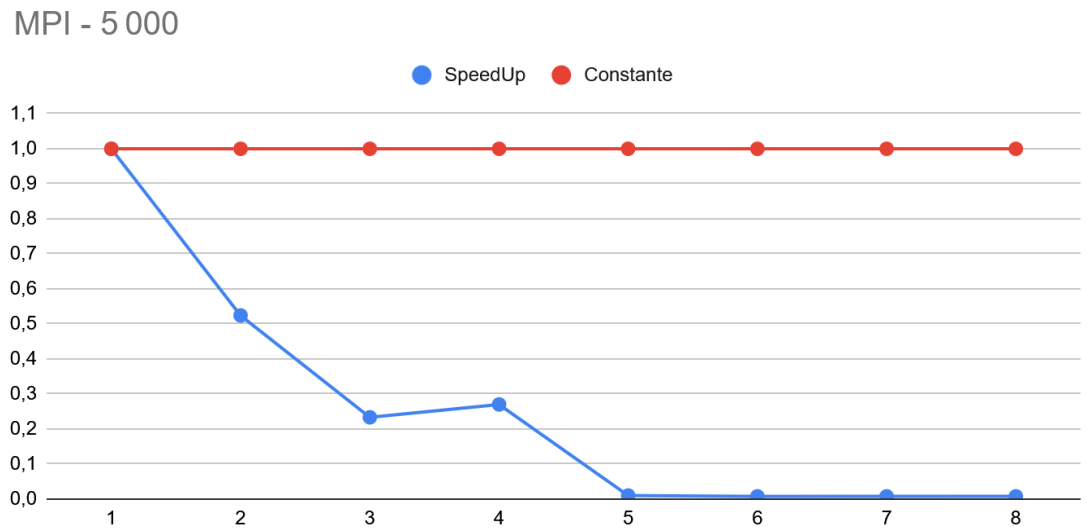
Pour réaliser cette expérience, on mesurera le temps d'exécution pour la recherche de nombre premier jusqu'à 5 000 en utilisant 1 à 8 cœurs.
Pour la scalabilité faible, la taille du problème augmente proportionnellement aux ressources, la limite de recherche est multipliée par le nombre de cœurs soit jusqu'à 40 000 pour 8 cœurs.

On effectuera à chaque fois 5 expériences similaires dont on gardera la valeur médiane de ces expériences.

2. Résultats

Après avoir réalisé nos expériences, nous obtenons le tableau et le graphique suivant :

nb tot	5 000	10 000	15 000	20 000	25 000	30 000	35 000	40 000
nb processus p	1	2	3	4	5	6	7	8
constante k	1	1	1	1	1	1	1	1
Sp	1	0,5239623272	0,2336290577	0,2700052848	0,0102107131	0,007814202118	0,008150104827	0,008102287964
Tp	284777683	543507936	1218930923	1054711515	27890087611	36443603415	34941597569	35147810625
Res1	307873716	527864737	1287954092	1050846787	27890087611	38386156744	33743640944	35027247721
Res2	284777683	603763334	1142034231	1062259804	25720564865	35933877817	33271798283	38088800973
Res3	281317856	557651004	1218930923	1054711515	23701320936	35108517985	34941597569	31821770953
Res4	282889541	541192076	1188525367	1080151339	28907445159	36443603415	35019655046	35147810625
Res5	289955551	543507936	1346419348	1003655679	29201995437	49602050049	40435072314	36162068955



Sur ce graphique :

- La courbe Constante désigne la courbe attendue pour un programme avec une scalabilité faible optimale.
- La courbe SpeedUp désigne la courbe obtenue après l'expérience menée sur le code MPI.

3. Analyse des Résultats

Observation : Contrairement aux autres programmes, la scalabilité faible n'est pas du tout validée, même sur le Raspberry 4.

- **De 1 à 4 processus :** La courbe décroche immédiatement. Dès le passage à 2 processus, le Speedup ne se maintient pas à 1, mais chute fortement. Il continue ensuite de baisser pour stagner aux alentours de 0,25 avec 3 et 4 processus.
- **À partir de 5 processus :** La performance s'effondre ici aussi totalement avec l'arrivée des Raspberry 0, atteignant un Speedup proche de 0,01.

Cette observation s'explique par le fait que la complexité du calcul augmente avec la taille des nombres. Vérifier si un grand nombre est premier demande exponentiellement plus de ressources que pour les petits nombres. Ce qui explique donc une scalabilité faible aussi faible

V. Analyse de performance de Simpson

A. Analyse de la scalabilité forte

1. Objectif

Mesurer si le programme de calcul d'intégrale par la méthode de Simpson s’exécute plus vite quand on augmente le nombre de processeurs sans changer la taille du problème.

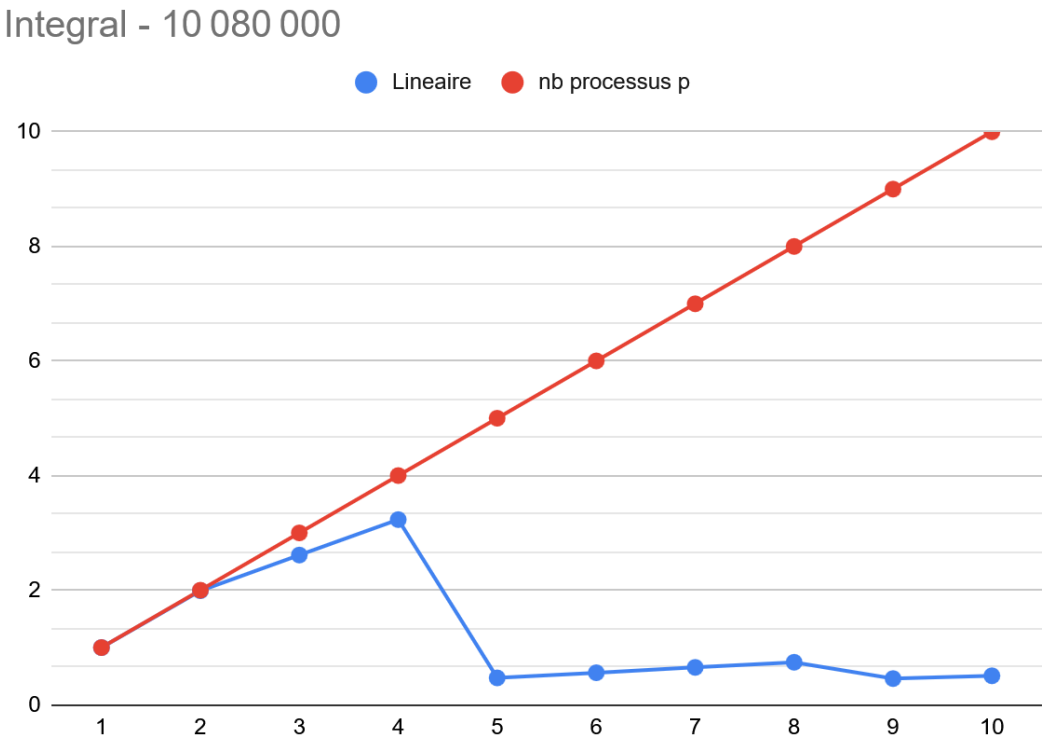
Pour réaliser cette expérience, on mesurera le temps d'exécution de 10 080 000, 100 800 000 et 1 008 000 000 exécutions en utilisant 1 à 10 threads.

On effectuera à chaque fois 5 expériences similaires dont on gardera la valeur médiane de ces expériences.

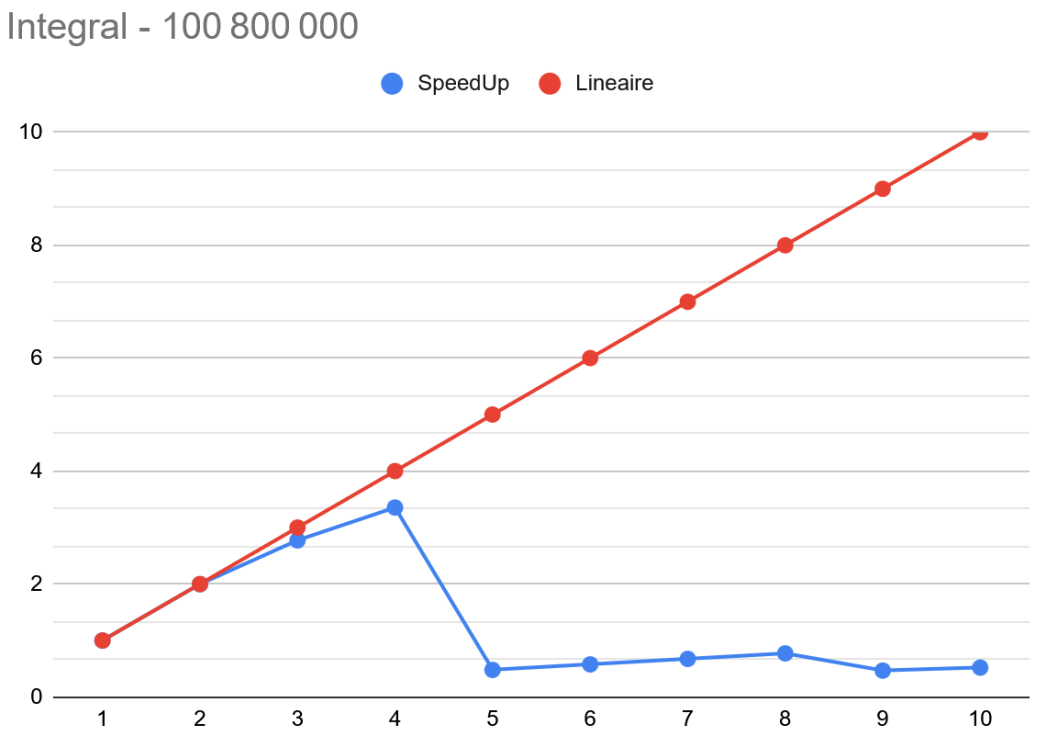
2. Résultats

Après avoir réalisé nos expériences, nous obtenons les tableaux et les graphiques suivants :

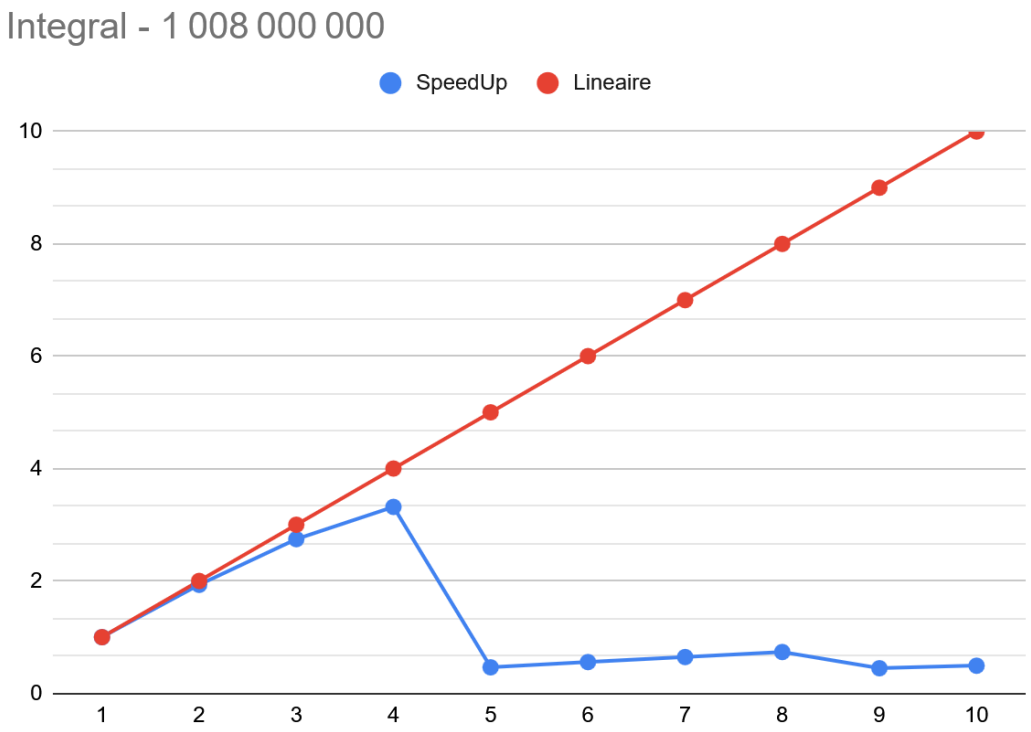
nb tot	10 080 000	10 080 000	10 080 000	10 080 000	10 080 000	10 080 000	10 080 000	10 080 000	10 080 000	10 080 000
nb tot par wk	10 080 000	5 040 000	3 360 000	2 520 000	2 016 000	1 680 000	1 440 000	1 260 000	1 120 000	1 008 000
nb processus p	1	2	3	4	5	6	7	8	9	10
Sp	1	1,990090832	2,614095062	3,233009669	0,4714527957	0,5592471336	0,6551661599	0,7413998739	0,4595365806	0,5068093513
Tp	360428698	181111682	137878956	111483953	764506439	644489129	550133264	486146155	784330809	711172154
Res1	424212214	243880654	199862808	190443338	958870624	841138774	772329089	679250746	122887520	1095529526
Res2	376387503	190880333	141521509	132806291	757220723	653324994	563093750	492975302	785957162	717291347
Res3	360428698	180921683	133510061	111397991	774177665	637807499	549026794	486146155	763590557	708058356
Res4	360052147	180826666	137878956	111483953	764506439	644489129	550133264	462043940	784330809	710258795
Res5	360183237	181111682	133595004	111403231	740911218	617265762	534357808	462080642	787599107	711172154



10 001 880	100 800 000	100 800 000	100 800 000	100 800 000	100 800 000	100 800 000	100 800 000	100 800 000	100 800 000	100 800 000
nb tot par wk	100 800 000	50 400 000	33 600 000	25 200 000	20 160 000	16 800 000	14 400 000	12 600 000	11 200 000	10 080 000
nb processus p	1	2	3	4	5	6	7	8	9	10
Sp	1	1,996236963	2,773030455	3,352289954	0,4795127871	0,5765387757	0,6726250606	0,7695904398	0,467055596	0,5190458412
Tp	3644878227	1825874530	1314402523	1087280121	7601211741	6322000151	5418885558	4736127216	7803949376	7022266509
Res1	3772369633	1925739946	1419979729	1178286840	7817014299	6553015432	5801798389	4946861207	8214699719	7461276791
Res2	3737131318	1893957427	1389433584	1107039940	7803270064	6318212115	5417288203	4736127216	7799154826	7017615707
Res3	3588594568	1825874530	1314221210	1071981068	7573070604	6317468476	5406289936	4734805842	7803949376	7026474482
Res4	3644878227	1797372956	1314402523	1087280121	7594879457	6325624109	5420136747	4739784770	8041491965	7022266509
Res5	3588036002	1798539535	1313921491	1086988661	7601211741	6322000151	5418885558	4735027526	7803804986	7011333848



nb tot	1 008 000 000	1 008 000 000	1 008 000 000	1 008 000 000	1 008 000 000	1 008 000 000	1 008 000 000	1 008 000 000	1 008 000 000	1 008 000 000
nb tot par wk	1 008 000 000	504 000 000	336 000 000	252 000 000	201 600 000	168 000 000	144 000 000	126 000 000	112 000 000	100 800 000
nb processus p	1	2	3	4	5	6	7	8	9	10
Sp	1	1,930195655	2,742047102	3,315720634	0,461289633	0,5547731343	0,6437697038	0,7335235912	0,4450499591	0,4924050916
Tp	35935889301	18617744379	13105496721	10838032894	77903093256	64775828310	55821032100	48990775120	80745742287	72980336550
Res1	37284347687	18968184722	13512036970	10911949986	78132612217	65303716316	55741575340	48881985515	80745742287	72817219653
Res2	37274295682	18617744379	13391683382	10838032894	77985892594	64632977117	55821032100	49120456780	81120556600	73150445880
Res3	35874027605	18909050875	13105496721	10843407912	77862650131	64775828310	56102458990	48560112340	79980443210	72450991200
Res4	35935889301	17994574401	13100956796	10693456969	77707883584	64632027964	55430112450	48990775120	80540112990	72980336550
Res5	35913480003	18006421683	13101544901	10763302596	77903093256	64899017661	55912347800	49230558900	81350887700	73401128900



Sur ces graphiques :

- La courbe Linéaire désigne la courbe attendue pour un programme avec une scalabilité forte optimale.
- La courbe SpeedUp désigne la courbe de scalabilité forte obtenue après l'expérience menée sur le code de calcul d'intégrale.

3. Analyse des Résultats

Observation : On retrouve des résultats très proches du code de Monte Carlo : Peu importe la taille du problème (10 puissance 6, 10 puissance 7 ou 10 puissance 8), la courbe a toujours la même forme.
De 1 à 4 processus : Le Speedup monte de façon linéaire et atteint 3,5 au 4e processus presque partout, ce qui est le résultat idéal.

À partir de 5 processus : Les courbes chutent brutalement. Le Speedup tombe en dessous de 0,5, ce qui signifie que le calcul devient plus lent qu'avec un seul processeur.

B. Analyse de la scalabilité faible

1. Objectif

Mesurer si le programme de calcul d'intégrale par la méthode de Simpson garde le même temps d'exécution quand on augmente le nombre de threads tout en augmentant la taille totale du problème proportionnellement.

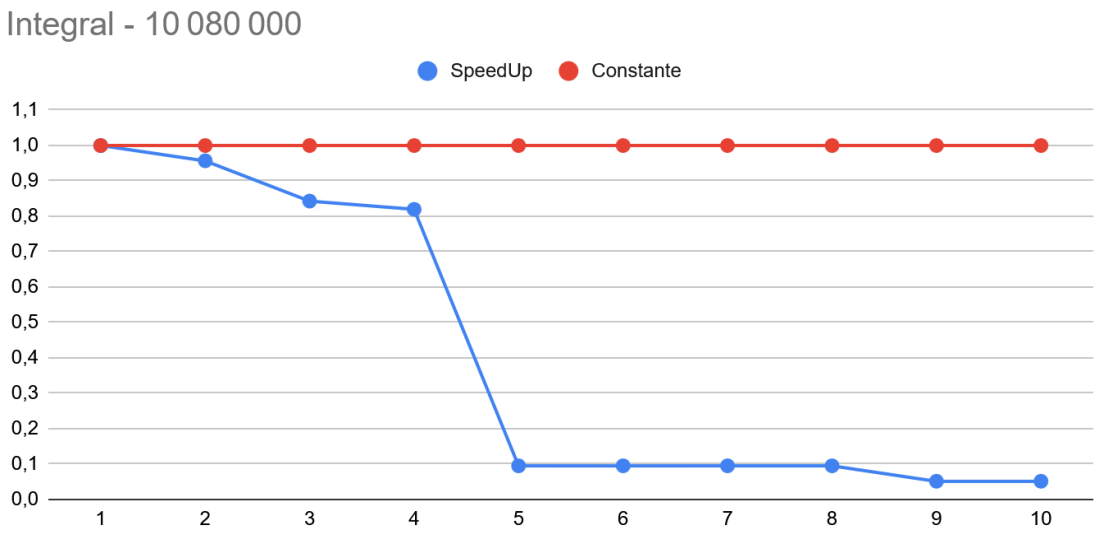
Pour réaliser cette expérience, on mesurera le temps d'exécution pour 10 080 000 et 100 800 000 exécutions par thread. On effectuera cette expérience avec 1 à 10 threads.

On effectuera à chaque fois 5 expériences similaires dont on gardera la valeur médiane de ces expériences.

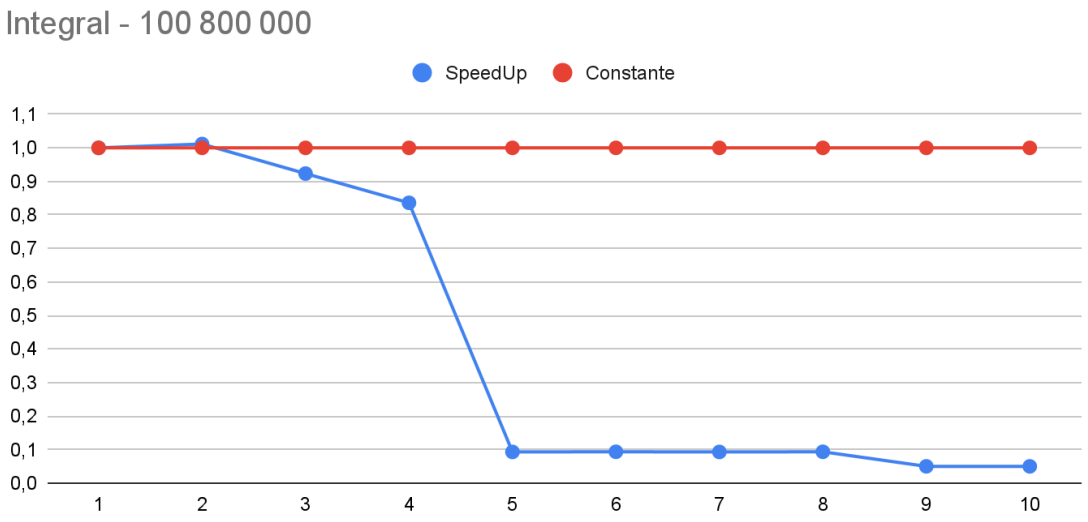
2. Résultats

Après avoir réalisé nos expériences, nous obtenons les tableaux et les graphiques suivants :

nb tot	10 080 000	20 160 000	30 240 000	40 320 000	50 400 000	60 480 000	70 560 000	80 640 000	90 720 000	100 800 000
nb processus p	1	2	3	4	5	6	7	8	9	10
constante k	1	1	1	1	1	1	1	1	1	1
Sp	1	0,9565437648	0,8428266442	0,8197209367	0,09519185874	0,09508776421	0,09517595537	0,09521931629	0,0513416157	0,05136426386
Tp	360708905	377096081	427975204	440038663	3789283136	3793431342	3789916304	3788190454	7025663296	7022565455
Res1	489861624	430732806	467996833	529025183	3993666761	3979067590	3993840918	4001127981	7412268969	7443425313
Res2	377763916	377096081	477637224	452798193	3787482485	3999096513	3791426670	3792393071	7014551626	7022565455
Res3	360095523	361319894	395488156	430883139	3800467210	3790683625	3789916304	3778615866	7025663296	7023730674
Res4	359991783	360298495	427975204	430810529	3776317476	3793431342	3777858601	3788190454	6988774626	7014164303
Res5	360708905	405146391	395164844	440038663	3789283136	3790873374	3784415051	3777776422	7294291884	7019050052



nb tot	100 800 000	201 600 000	302 400 000	403 200 000	504 000 000	604 800 000	705 600 000	806 400 000	907 200 000	1 008 000 000
nb processus p	1	2	3	4	5	6	7	8	9	10
constante k	1	1	1	1	1	1	1	1	1	1
Sp	1	1,011438508	0,9233280701	0,836083157	0,09307062484	0,09345089795	0,09299777782	0,09343311786	0,05006965154	0,04999134927
Tp	3631915982	3590842107	3933505435	4343965013	39023225516	38864431073	39053793188	38871826878	72537272983	72650889300
Res1	3772966352	3784739357	4138216738	4462972958	39023225516	38864431073	39053793188	38871826878	72537272983	72582284738
Res2	3720886962	3725219431	4074376616	4389873647	38910445200	38920115500	39120456000	38950441200	72650114500	72700451200
Res3	3590242258	3589434968	3933379855	4287811370	39150678990	38790554120	38950112880	38790115660	72420889100	73490112880
Res4	3631915982	3590842107	3933018788	4343965013	38990112340	38840998800	39080667120	38840992300	72590336220	72650889300
Res5	3592033695	3589660050	3933505435	4294338955	39080221100	39010223450	38990225400	39010558990	72380775400	71390221440



Sur ces graphiques :

- La courbe Constante désigne la courbe attendue pour un programme avec une scalabilité faible optimale.
- La courbe SpeedUp désigne la courbe de scalabilité faible obtenue après l'expérience menée sur le code de calcul d'intégrale.

3. Analyse des Résultats

Observation :
De 1 à 4 processus : Le Speedup diminue légèrement, passant de 1 à environ 0,82, ce qui démontre une scalabilité faible qui n'est pas optimale, mais qui reste acceptable.

À partir de 5 processus : On constate ici aussi une chute brutale lors de l'arrivée des rpi0. Le Speedup tombe à environ 0,09 et reste stable à ce niveau bas.

À partir de 9 processus : Le speedUp diminue encore et atteint 0,05 dans toutes les expériences

VI. Conclusion

Au terme de ces expériences, nous constatons que les résultats de Monte Carlo, Simpson et MPI suivent tous une tendance similaire divisée en trois phases distinctes. Cependant, une exception importante a été relevée concernant la scalabilité faible de MPI.

Voici l'analyse globale de nos observations

- **De 1 à 4 processus :** C'est la configuration la plus efficace. Le calcul se fait uniquement sur le **Master** (RPi 4). On profite de ses 4 cœurs puissants (1.8 GHz) sans avoir besoin d'envoyer des données par le réseau.
    - Pour Monte Carlo et Simpson : La scalabilité forte et faible est validée. Le Speedup est linéaire et l'efficacité est maintenue
    - Contrairement aux autres, MPI ne maintient pas ses performances lors des tests de scalabilité faible. Cela s'explique par la nature mathématique du problème : vérifier de très grands nombres premiers demande beaucoup plus de calculs que pour les petits nombres. Ce qui fait croitre exponentiellement la demande en ressources lorsqu'on augmente la taille du problème.
- **De 5 à 8 processus :** Dès l'ajout du 5e processus, les courbes de performance de tous les programmes décrochent fortement. C'est le moment où le calcul devient distribué. Cette chute s'explique par deux raisons :
    1. **Le réseau :** Le Master doit envoyer des données aux autres machines, ce qui prend du temps.
    2. **La différence de puissance :** Les workers 5 à 8 tournent sur des **Raspberry Pi 0**. Les cœurs de ces Rpi0 (1.0 GHz) sont beaucoup moins puissants que ceux du Rpi4 (1.8 GHz). Comme le programme doit attendre que tout le monde ait fini, le temps global d'exécution augmente fortement.
- **9 et 10 processus :** Les performances continuent de se dégrader. En assignant plusieurs tâches aux mêmes cœurs, on augmente le temps de calcul, ce qui fait encore plus baisser la courbe.

Bilan : L'expérience montre que nos algorithmes sont globalement performants sur le Raspberry 4. Mais le passage en calcul distribué n'apporte aucun gain et entraine même une perte importante des performances. En effet, l''écart de puissance trop important entre le Raspberry 4 et les Raspberry 0 crée un goulot d'étranglement qui annule les bénéfices du parallélisme sur ce cluster.