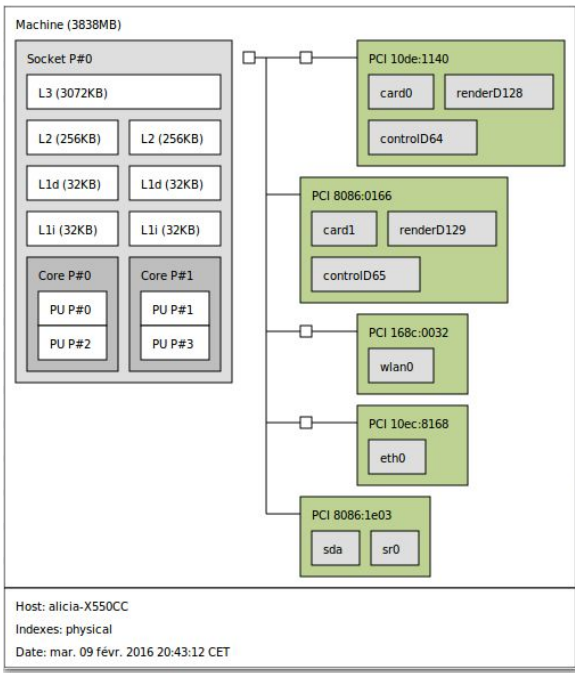
Abonnenc Alicia
Bonhoure Gilles

RICM3

Compte-rendu MN TP01

Présentation de la machine de test numéro 1



Nous avons travaillé sur une machine dotée d'un processeur doté d'une fréquence à 1.8GHz, et d'un cache L3 de 3072KB d'espace. Il est important de noter ce genre de détails, car un programme ne tournera pas de la même façon en fonction des performances d'une machine. Par exemple, dans le cadre de ce TP, l'espace cache est important pour tester les différentes façons de calculer, en dépassant ou non la taille de cache (et donc en accédant ou non à la mémoire en dehors du processeur). La fréquence du processeur définit également le nombre d'opérations effectuées, et influe donc sur le nombre de flops (*float operation per second*) lors de l'exécution d'un programme. Nous avons utilisé la version du compilateur gcc 4.8.4.

Présentation du travail

Le but de ce TP est de se familiariser à l'optimisation de code, et d'analyser le comportement d'un programme C en fonction des options de compilations, et du contenu du programme en lui-même.
Les résultats devraient confirmer qu'un programme avec moins d'appels à la mémoire sur plus rapides, et requièrent moins d'opérations. Il devrait également apparaître qu'une optimisation de la compilation est fortement conseillée. Pour finir, le dépassement du cache L3 du processeur est censé prendre plus longtemps, et donc être beaucoup moins efficace qu'un programme sans dépassement.

Protocole de test

Pour cela, nous avons donc réalisé le code d'un **produit scalaire** et d'une **multiplication de vecteurs**, sur des **float** et des **double**. Nous avons également fait différents tests en fonction de la **mémoire cache du processeur** pour voir s'il y avait une grande différence de temps de calcul lors d'un **dépassement**, ainsi que **différentes** méthodes de **compilations**, pour obtenir différents résultats avec gcc.

Résultats des tests et opérations flottantes par seconde sur la machine 1

N = 100 000 et ITER = 100 000

Sans dépassement de cache. Sans optimisation.

Opération	Type	Temps en ms	MFLOPS	Version classique	Boucle déroulée
Addition	Floats	28050	356	X	
		27335	365		X
	Double	28458	351	X	
		27805	359		X
Multiplication	Floats	22464	445	X	
		19161	521		X
	Double	22674	441	X	
		23987	416		X
Produit Scalaire	Floats	23003	869	X	
		22472	889		X
	Double	22493	889	X	
		21734	920		X

Sans dépassement de cache. optimisation -O2

Opération	Type	Temps en ms	MFLOPS	Version classique	Boucle déroulée
Addition	Floats	1172	850	X	
		958	1043		X
	Double	1580	632	X	
		1631	612		X
Multiplication	Floats	1120	892	X	
		763	1309		X
	Double	1251	799	X	
		1175	850		X
Produit Scalaire	Floats	0	inf	X	
		1674	1194		X
	Double	0	inf	X	
		1679	1190		X

Sans dépassement de cache. optimisation -O3

Opération	Type	Temps en ms	MFLOPS	Version classique	Boucle déroulée
Addition	Floats	6486	1541	X	
		10691	935		X
	Double	14043	712	X	
		21880	457		X
Multiplication	Floats	4710	2123	X	
		4663	2144		X
	Double	9460	1056	X	
		16368	610		X
Produit Scalaire	Floats	0	inf	X	
		0	inf		X
	Double	0	inf	X	
		0	inf		X

N = 400 000 et ITER = 1000

Avec dépassement de cache double. Sans optimisation.

Opération	Type	Temps en ms	MFLOPS	Version classique	Boucle déroulée
Addition	Floats	2444	327	X	
		2237	357		X
	Double	2687	297	X	
		2708	295		X
Multiplication	Floats	1862	429	X	
		1644	486		X
	Double	2005	398	X	
		2067	387		X
Produit Scalaire	Floats	1891	845	X	
		1838	870		X
	Double	1918	834	X	
		1832	873		X

Avec dépassement de cache double. optimisation -O2

Opération	Type	Temps en ms	MFLOPS	Version classique	Boucle déroulée
Addition	Floats	1056	757	X	
		1078	742		X
	Double	2554	313	X	
		2547	314		X
Multiplication	Floats	917	872	X	
		653	1224		X
	Double	1842	434	X	
		1800	444		X
Produit Scalaire	Floats	0	inf	X	
		1362	1174		X
	Double	0	inf	X	
		1417	1128		X

Sans dépassement de cache double. optimisation -O3

Opération	Type	Temps en ms	MFLOPS	Version classique	Boucle déroulée
Addition	Floats	893	895	X	
		880	908		X
	Double	2488	321	X	
		2653	301		X
Multiplication	Floats	500	1598	X	
		490	1629		X
	Double	1723	464	X	
		1831	436		X
Produit Scalaire	Floats	0	inf	X	
		0	inf		X
	Double	0	inf	X	
		0	inf		X

N = 800 000 et ITER = 1000

Avec dépassement de cache float. Sans optimisation.

Opération	Type	Temps en ms	MFLOPS	Version classique	Boucle déroulée
Addition	Floats	4758	336	X	
		4680	341		X
	Double	5512	290	X	
		5450	293		X
Multiplication	Floats	3738	427	X	
		3225	496		X
	Double	4381	365	X	
		4370	366		X
Produit Scalaire	Floats	3723	859	X	
		3740	855		X
	Double	3800	841	X	
		3992	801		X

Avec dépassement de cache float. optimisation -O2

Opération	Type	Temps en ms	MFLOPS	Version classique	Boucle déroulée
Addition	Floats	2694	593	X	
		2681	596		X
	Double	5375	297	X	
		5439	294		X
Multiplication	Floats	2246	712	X	
		2109	758		X
	Double	4055	394	X	
		3990	400		X
Produit Scalaire	Floats	0	inf	X	
		2751	1162		X
	Double	0	inf	X	
		3195	1001		X

Sans dépassement de cache float. optimisation -O3

Opération	Type	Temps en ms	MFLOPS	Version classique	Boucle déroulée
Addition	Floats	2860	559	X	
		2751	581		X
	Double	5506	290	X	
		5324	300		X
Multiplication	Floats	1790	893	X	
		1777	900		X
	Double	3945	405	X	
		4068	393		X
Produit Scalaire	Floats	0	inf	X	
		0	inf		X
	Double	0	inf	X	
		0	inf		X

Conclusion

En conclusion, on remarque que dépasser la taille du cache L3, et donc devoir accéder à la mémoire de l'ordinateur ne change sensiblement rien en temps de calcul, et en nombre de flops. De plus, on remarque bien que les différentes méthodes de compilation influent beaucoup sur l'exécution finale du programme. En effet, l'optimisation lors de la compilation n'est pas négligeable, et permet parfois même d'éviter les doublons dans le code, en vérifiant les résultats de calculs… On remarque également que de manière générale, le nombre d'opérations flottantes dans un temps donné est beaucoup moins important quand on travaille sur une double précision que sur des flottants.
Toutes les hypothèses que l'on avait énoncé sont donc vérifiées, bien que pour certaines telles que le temps d'accès à la mémoire lors du dépassement du cache, la différence ne soit pas aussi notable que ce que l'on pensait.