IF5-PAR 1 : programmation Cell

Thibault Jacquemin

Mathieu Marleix

Année 2012-2013

Contenu

TP1

Organisation des sources

Un fichier C (*simple.c*) accompagné d'un Makefile général gérant les threads SPU.

Puis un sous-dossier contenant un fichier C (*simple_spu.c*) et un Makefile pour créer le programme lancé par un thread SPU.

Compilation

Voici dans l'ordre les étapes réalisés lors d'un appel au Makefile général :

- Le compilateur se place dans le sous-dossier spu et appelle le sous-makefile pour SPU ;
- Il appelle la librairie extérieure *libc.a* contenue dans le SDK Cell ;
- Sont créés la librairie locale *lib_simple_spu.a* et le programme pour thread SPU *simple_spu* ;
- Le compilateur revient au dossier principal et crée le programme PPU *simple* à l'aide de la librairie locale créée juste avant.

Le programme *simple* crée autant de threads qu'il y a de cœurs visibles (8 cœurs physiques pour chacun des deux processeurs Cell présents sur la lame).

Chaque thread imprime alors le message suivant :

> Hello Cell <id_thread>

Sortie console :

```
[user2@blade09 tp1]$ ./simple
Hello Cell (0x1818008)
Hello Cell (0x1818280)
Hello Cell (0x1818518)
Hello Cell (0x1818790)
Hello Cell (0x1818a08)
Hello Cell (0x1818c80)
Hello Cell (0x1818ef8)
Hello Cell (0x1819170)
Hello Cell (0x18193e8)
Hello Cell (0x1819660)
Hello Cell (0x18198d8)
Hello Cell (0x1819b50)
Hello Cell (0x1819dc8)
Hello Cell (0x181a040)
Hello Cell (0x181a2b8)
Hello Cell (0x181a530)
```

TP2

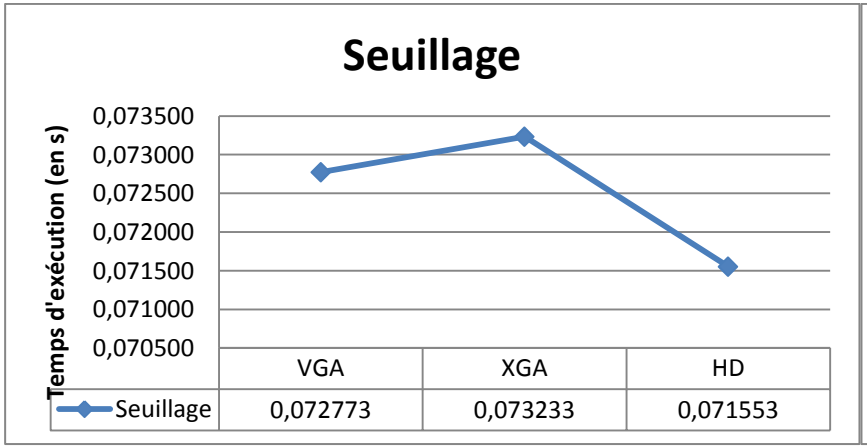
Analyse du programme

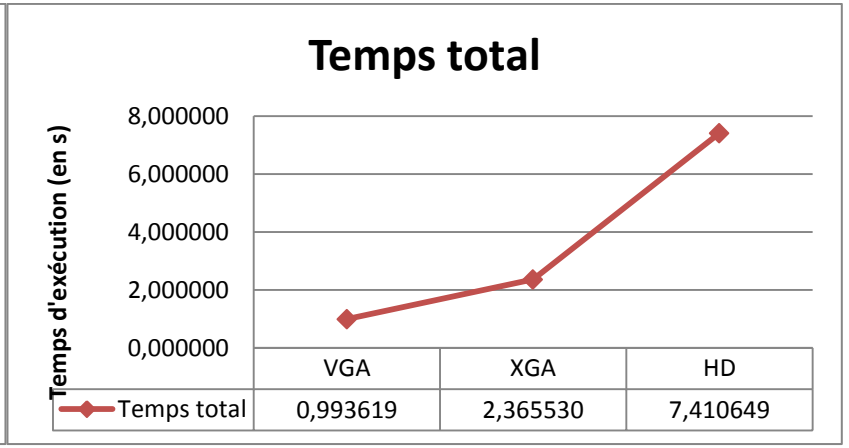
Le programme traite des images en niveaux de gris.

L'image cible est allouée statiquement et chargée en mémoire dans le tas, puis est seuillée pixel par pixel, et enfin stockée dans une image de sortie.

Tests

TP2	VGA				XGA				HD			
	Chargement	Seuillage	Sauvegarde	Temps total	Chargement	Seuillage	Sauvegarde	Temps total	Chargement	Seuillage	Sauvegarde	Temps total
Itération 1	0,395029	0,072778	0,485696	0,953503	0,964834	0,073202	1,233609	2,271645	3,068581	0,071124	3,772062	6,911767
Itération 2	0,376642	0,072671	0,479871	0,929184	0,967325	0,073102	1,244019	2,284446	3,087431	0,071366	3,713196	6,871993
Itération 3	0,396134	0,072791	0,492866	0,961791	0,968133	0,073089	1,247959	2,289181	3,087825	0,071480	3,721348	6,880653
Itération 4	0,376717	0,072921	0,481613	0,931251	0,984547	0,073947	1,366616	2,425110	3,123243	0,071269	3,714559	6,909071
Itération 5	0,378918	0,072714	0,482009	0,933641	1,002152	0,073140	1,233132	2,308424	3,089732	0,071245	3,713594	6,874571
Itération 6	0,383738	0,072791	0,499363	0,955892	1,466143	0,073077	1,233879	2,773099	4,006834	0,071309	3,731247	7,809390
Itération 7	0,921484	0,072803	0,481068	1,475355	0,988986	0,073152	1,236917	2,299055	3,097918	0,071257	3,720436	6,889611
Itération 8	0,378786	0,072916	0,481214	0,932916	1,030983	0,073422	1,230525	2,334930	4,548370	0,071173	3,770669	8,390212
Itération 9	0,378856	0,072706	0,480903	0,932465	0,991388	0,073175	1,232973	2,297536	3,093886	0,071327	4,767216	7,932429
Itération 10	0,377081	0,072643	0,480467	0,930191	1,066008	0,073025	1,232841	2,371874	4,231258	0,073979	4,331556	8,636793
Moyenne	0,436339	0,072773	0,484507	0,993619	1,043050	0,073233	1,249247	2,365530	3,443508	0,071553	3,895588	7,410649





TP3

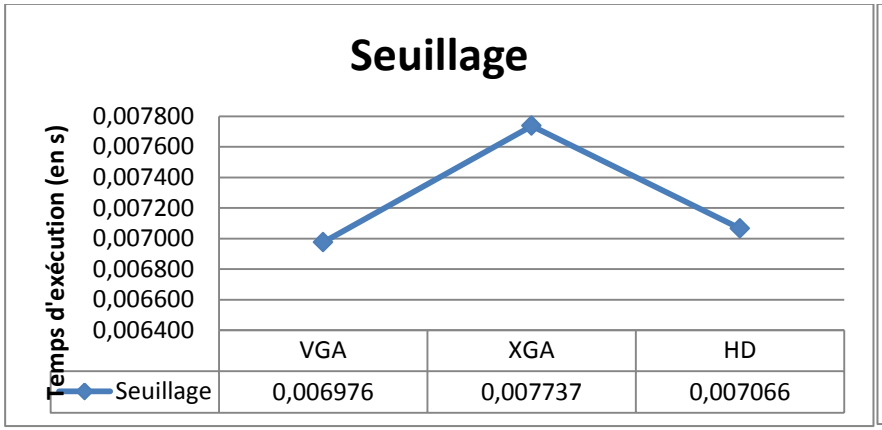
Analyse du programme

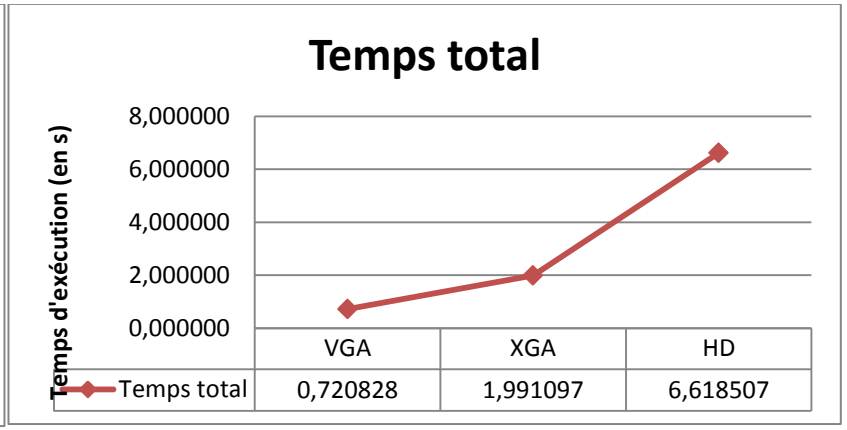
Le programme fonctionne de la même manière que celui du tp2. Cependant, l'image est découpée en mémoire par bloc de 16 bytes et le seuillage ne se fait pas pixel par pixel mais par également par vecteur de 16 bytes.

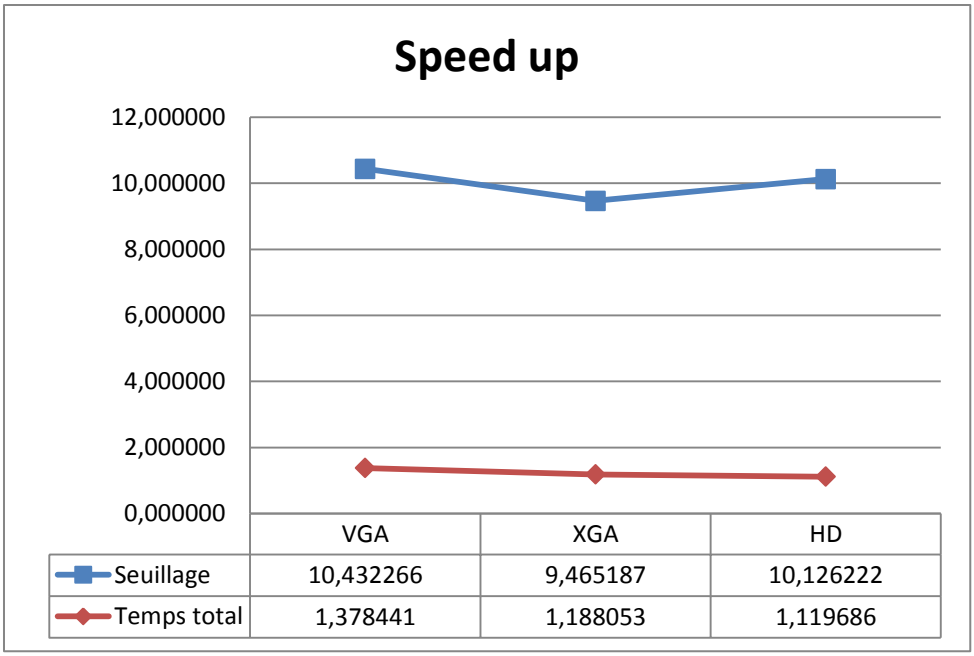
Si les opérations sur un même vecteur s'effectuent toutes simultanément, le gain attendu sur le seuillage serait donc de 16.

Tests

TP3	VGA				XGA				HD			
	Chargement	Seuillage	Sauvegarde	Temps total	Chargement	Seuillage	Sauvegarde	Temps total	Chargement	Seuillage	Sauvegarde	Temps total
Itération 1	0,334059	0,006992	0,405492	0,746543	0,819996	0,006873	0,976937	1,803806	2,881503	0,006695	3,460414	6,348612
Itération 2	0,331219	0,007215	0,383175	0,721609	0,831901	0,006971	1,009630	1,848502	2,894678	0,006763	3,492043	6,393484
Itération 3	0,322791	0,006889	0,380801	0,710481	0,830076	0,006936	0,981919	1,818931	2,895654	0,006744	3,433029	6,335427
Itération 4	0,323053	0,007021	0,392181	0,722255	0,827074	0,006932	1,005740	1,839746	2,890322	0,006743	3,458059	6,355124
Itération 5	0,330358	0,006837	0,381014	0,718209	0,823102	0,006921	0,979889	1,809912	4,703043	0,009628	4,090204	8,802875
Itération 6	0,323204	0,006992	0,381571	0,711767	0,993506	0,008782	1,162107	2,164395	2,896617	0,006811	3,470442	6,373870
Itération 7	0,324574	0,006918	0,380816	0,712308	0,909719	0,008932	1,161319	2,079970	2,889634	0,006846	3,472130	6,368610
Itération 8	0,331359	0,006894	0,392965	0,731218	0,911545	0,008837	1,178283	2,098665	2,909973	0,006836	3,461287	6,378096
Itération 9	0,322518	0,006893	0,381902	0,711313	1,337106	0,007361	1,018561	2,363028	2,904025	0,006738	3,467790	6,378553
Itération 10	0,323727	0,007107	0,391742	0,722576	0,913826	0,008826	1,161366	2,084018	2,959970	0,006857	3,483589	6,450416
Moyenne	0,326686	0,006976	0,387166	0,720828	0,919785	0,007737	1,063575	1,991097	3,082542	0,007066	3,528899	6,618507
Speed up	1,335650	10,432266	1,251420	1,378441	1,134015	9,465187	1,174573	1,188053	1,117100	10,126222	1,103910	1,119686





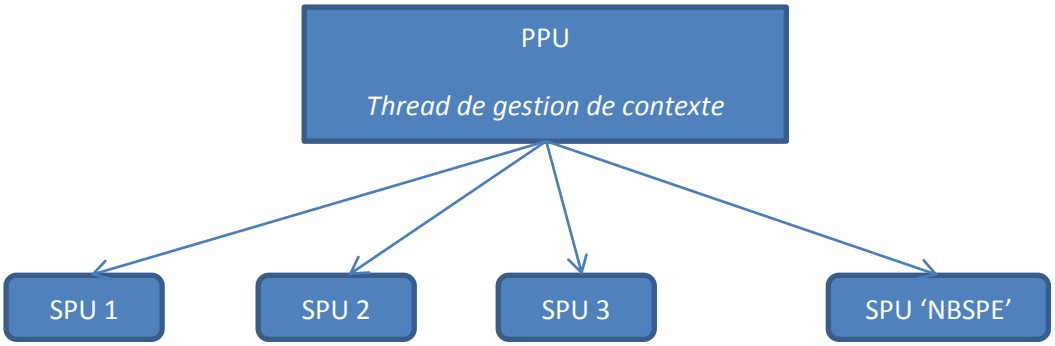


Le speed up du seuillage est inférieur à ce qui était attendu. Cela vient notamment du fait que les opérations mises en œuvre (comme celles sur les vecteurs) sont plus coûteuses.

De plus, le seuillage ne représentant qu'une infime partie du temps d'exécution du programme, le speed up total est faible, cela se voyant au fur et à mesure que la taille de l'image traitée augmente (et donc les temps de chargement/sauvegarde).

TP4

Organigramme



Le programme PPU crée un thread par SPU, charge le programme à exécuter et le lance.

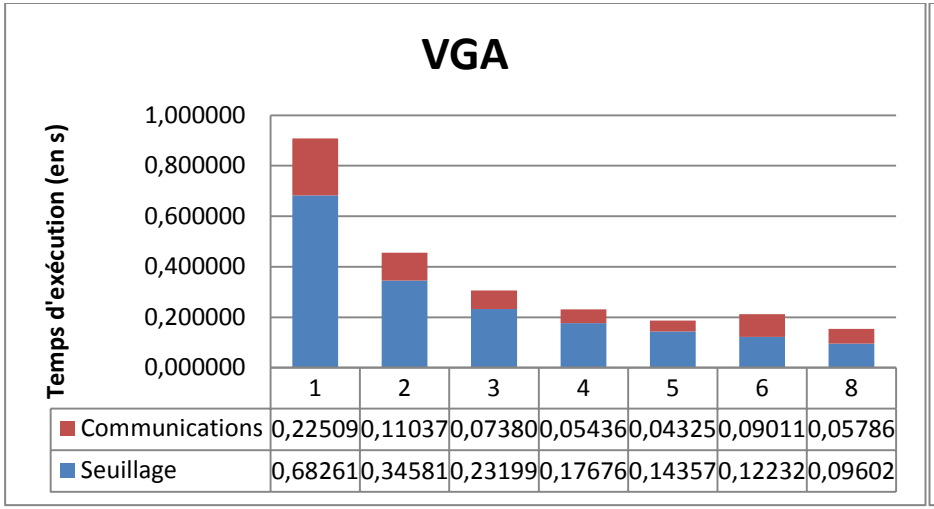
Mesure des surcoûts de communication

Tout d'abord, le but de ces tests étant de tester les gains effectués en modifiant le nombre de processeurs traitant le seuillage, nous ne mesurons que la partie relative aux threads SPU. Cela exclut donc les temps de chargement et de sauvegarde de l'image dans le programme PPU.

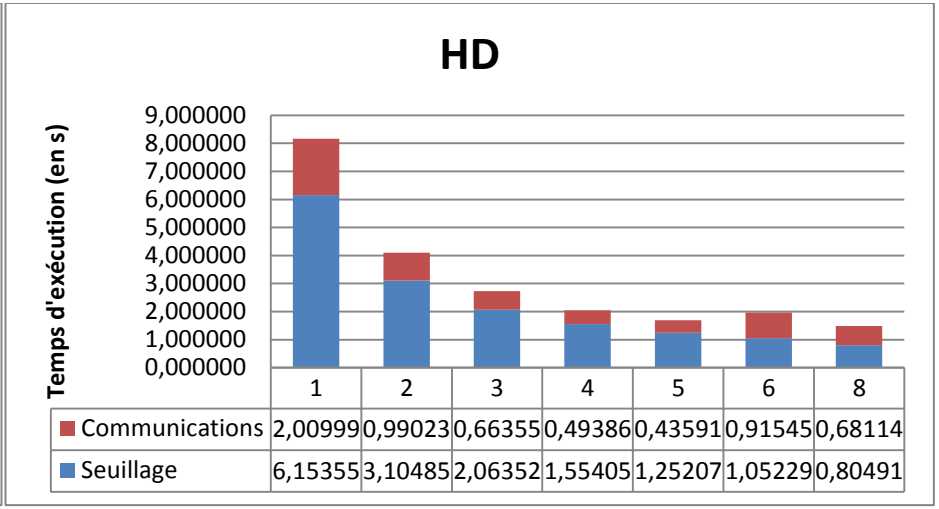
Pour mesurer les surcoûts de communication entre cœurs, nous relançons le programme en enlevant les lignes effectuant le seuillage dans le script SPU. Le surcoût sera obtenu par soustraction avec les temps précédemment obtenus.

Tests

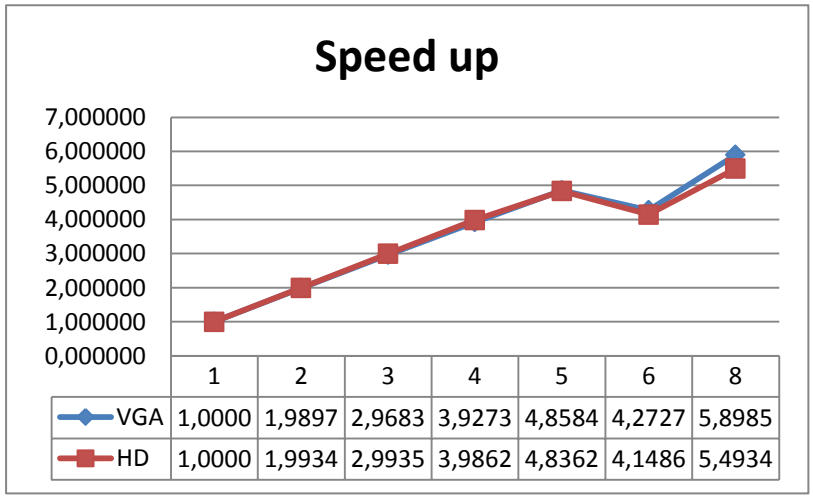
TP4	Exécution avec calcul du seuillage													
	VGA							HD						
Nb de SPU(s)	1	2	3	4	5	6	8	1	2	3	4	5	6	8
Itération 1	0,907857	0,456191	0,305805	0,231100	0,186931	0,226105	0,123122	8,163986	4,094945	2,727220	2,047537	1,640463	1,967963	1,486050
Itération 2	0,907651	0,456264	0,305813	0,231087	0,186846	0,225705	0,122963	8,163377	4,095067	2,726897	2,048286	1,641404	1,967607	1,485964
Itération 3	0,907615	0,456170	0,305812	0,231168	0,186864	0,225887	0,174562	8,163474	4,094772	2,727046	2,048096	1,876318	1,967649	1,486200
Itération 4	0,907732	0,456154	0,305791	0,231158	0,186750	0,226753	0,174391	8,163387	4,095166	2,727221	2,048016	1,640719	1,968785	1,486175
Itération 5	0,907683	0,456155	0,305744	0,231119	0,186765	0,157757	0,174394	8,163509	4,095495	2,727001	2,047686	1,641037	1,966776	1,485900
Moyenne	0,907708	0,456187	0,305793	0,231126	0,186831	0,212441	0,153886	8,163547	4,095089	2,727077	2,047924	1,687988	1,967756	1,486058
Speed up	1,000000	1,989772	2,968373	3,927321	4,858437	4,272743	5,898556	1,000000	1,993497	2,993515	3,986254	4,836258	4,148658	5,493425
TP4	Exécution sans calcul du seuillage													
	VGA							HD						
Nb de SPU(s)	1	2	3	4	5	6	8	1	2	3	4	5	6	8
Itération 1	0,682551	0,345715	0,231964	0,176588	0,143593	0,122241	0,096045	6,154781	3,104508	2,063552	1,553118	1,251806	1,052017	0,802363
Itération 2	0,682711	0,346406	0,232081	0,176337	0,143632	0,122150	0,095994	6,153551	3,105859	2,063195	1,553422	1,252913	1,052519	0,802591
Itération 3	0,682547	0,345712	0,231947	0,176570	0,143627	0,122144	0,096019	6,153313	3,104645	2,062970	1,555744	1,252558	1,052205	0,803381
Itération 4	0,682552	0,345704	0,232035	0,176307	0,143542	0,122700	0,095993	6,153078	3,104323	2,063362	1,554040	1,251564	1,052390	0,802689
Itération 5	0,682713	0,345522	0,231926	0,178025	0,143491	0,122380	0,096068	6,153051	3,104959	2,064525	1,553957	1,251520	1,052355	0,813535
Moyenne	0,682615	0,345812	0,231991	0,176765	0,143577	0,122323	0,096024	6,153555	3,104859	2,063521	1,554056	1,252072	1,052297	0,804912
Surcoût des communications	0,225093	0,110375	0,073802	0,054361	0,043254	0,090118	0,057863	2,009992	0,990230	0,663556	0,493868	0,435916	0,915459	0,681146
	24,8%	24,2%	24,1%	23,5%	23,2%	42,4%	37,6%	24,6%	24,2%	24,3%	24,1%	25,8%	46,5%	45,8%



	1	2	3	4	5	6	8
Communications	0,22509	0,11037	0,07380	0,05436	0,04325	0,09011	0,05786
Seuillage	0,68261	0,34581	0,23199	0,17676	0,14357	0,12232	0,09602



	1	2	3	4	5	6	8
Communications	2,00999	0,99023	0,66355	0,49386	0,43591	0,91545	0,68114
Seuillage	6,15355	3,10485	2,06352	1,55405	1,25207	1,05229	0,80491



	1	2	3	4	5	6	8
VGA	1,0000	1,9897	2,9683	3,9273	4,8584	4,2727	5,8985
HD	1,0000	1,9934	2,9935	3,9862	4,8362	4,1486	5,4934

Le speed up est globalement le même pour les deux tailles d'image différentes.

Il est linéaire tant que nous utilisons 5 cœurs : cela signifie que les tâches à traiter sont réparties uniformément.

A partir de 6 cœurs, le speed up chute : la raison est que les coûts de communication sont trop importants par rapport à la quantité de données à traiter par chaque cœur.

En effet, à partir de ce moment, les communications représentent presque la moitié du temps d'exécution, quand elles n'en représentaient qu'un quart avant, et l'augmentation du nombre de cœurs n'est donc plus rentable pour les tailles d'image que nous avons ici.