Projet C-Wire - groupe A

En France, une centrale électrique peut alimenter des centaines, voire des milliers de stations intermédiaires, selon sa capacité et sa taille. La gestion de données d'un tel réseau est donc un véritable défi. C'est précisément cette problématique que nous cherchons à résoudre à travers ce projet.

Notre objectif est donc de développer un programme permettant de synthétiser et de traiter les données liées à un système de distribution d'électricité. Dans le cadre de ce projet, nous nous concentrerons sur les données de 5 centrales électriques, depuis la production jusqu'à la distribution aux différents acteurs du réseau.

Nous avons donc eu besoin de créer une partie de notre projet en shell, et une autre en langage c. Dans ce document, nous vous avons listé plusieurs points sur les “coulisses” de notre projet. Pour commencer, nous vous présentons comment nous avons divisé les tâches au sein du groupe. Ensuite, nous vous présentons notre planning. Enfin, nous vous listons les limitations fonctionnelles de notre projet.

A la fin, vous trouverez des exemples d’exécution de notre application expliqués. Nous pouvons retrouver les exemples expliqués dans le dossier ‘tests’ de notre rendu.

- **Répartition des tâches au sein du groupe**

Nous avons commencé par se mettre d’accord sur qui fait quoi en fonction de nos compétences. Après avoir bien compris la partie théorique du projet, Wiame et Sagana se sont lancées dans la construction de la partie en shell et Sanem sur le C. À l’approche de la date de rendu, nous avons toutes participé à la création de ce document PDF à travers un google doc partagé expliquant la répartition des tâches au sein du groupe, le planning de réalisation, les limites fonctionnelles de notre programme ainsi que quelques exemples d'exécution. De même, Wiame s’est chargée de la construction du ReadMe la dernière semaine.

Concernant notre manière de travailler, généralement, nous nous appelions afin de travailler ensemble et de partager nos idées concernant le programme. Wiame partageait son écran et nous travaillions ensemble. Lorsque l’une avançait sur une partie de code, elle attendait que nous en discutions lors de nos appels avant de l'ajouter sur notre github. Nous faisions tout cela à travers plusieurs plateformes comme FaceTime, Whatsapp ou par un canal Teams privé. Ensuite, en classe, nous essayions notre code généralement sur les machines pour être sûres d’être sur la bonne voie. Bien sûr, au début, nous nous concentrions sur la théorie et

nous posions nos questions au professeur. Ensuite, peu à peu, c'était beaucoup de débogage, surtout sur les 2 dernières semaines avant le rendu.

- **Planning**

Peu de temps après l'annonce du sujet, Wiame a rapidement créé un répertoire github et y a disposé les différents fichiers nécessaires et demandés (codeC, input, output, tests, tmp, graphs). Nous avons décidé de faire les choses dans cet ordre : le code c ; la fonction d'aide dans le shell ; le timer ; les vérifications de paramètres ; la vérification de compilation ; makefile ; tri ; bonus. De manière logique, nous nous sommes dit que lorsque tout ce qui est "contextuel" au projet, c'est-à-dire tout ce qui permettra un tri efficace qui est l'action principale de notre projet, sera fait, alors le tri qui est la dernière étape sera prêt à être fait sans problème sur le reste du programme. Nous nous sommes également dit que s'il nous restait du temps, nous pourrions faire le bonus, ce qu'on a fait à une semaine et demi du rendu environ. En effet, nous avons pu respecter ce planning.

Voici un résumé sur le temps de notre planning (sachant qu'on avait environ 1 mois pour faire ce projet) :

- Première semaine : création des fichiers, compréhension (théorie), partie c (début) // en même temps : fonction d'aide et vérifications des paramètres
- Deuxième semaine : code C (finitions) // vérifications + vérification compilation ; timer et début makefile
- Troisième semaine : tri
- Quatrième semaine : tri, readme, bonus et ce pdf

À noter que les commentaires se faisaient peu à peu, mais qu'ils ont été beaucoup réarrangés durant la dernière semaine pour que tout soit clair et propre.

- **Limitations fonctionnelles de notre programme**

- *Implémenté mais qui ne fonctionne pas correctement :*

→ Parfois, lorsque la capacité est trop grande, elle peut être modifiée par rapport à sa taille. Nous avons utilisé des long int, cependant cette erreur persiste. (Cela se voit avec la version v00, pas avec v25 car dans la version v25 nous n'avons pas de si grandes valeurs pour la capacité). Nous pensons que cela a à voir avec la commande awk qu'on utilise de manière récurrente pour notre tri dans la partie shell. Cette commande peut être sujette à des comportements indéfinis avec l'utilisation des %ld pour les long int…

→ Petit bug qui n'apparaît pas : dans la partie vérification de compilation dans le shell, nous remarquons que chaque lancement de programme compile de nouveau, même si l'exécutable existe… La vérification avec le if camoufle la redondance qui recommence à chaque fois sans lui.

- Pas implémenté :

Normalement, tout a été implémenté.

- **Exemples d'exécution de notre projet : (+ fichiers intermédiaires)**

P.S : Les exemples suivants sont tous obtenus en utilisant la version v25 des fichiers fournis

Traitement <u>hva comp 4</u> :

Pour ce traitement, comme celui d'un hvb comp ou d'un lv comp, nous n'avons pas besoin de fichier intermédiaire. Le traitement se fait de manière directe (qu'une centrale soit choisie comme élément de traitement par l'utilisateur ou non).

- hva_comp_4.csv

```
HVA-Stations:Capacity:TotalConsumption(companies)
321:175413406:67064396
288:189839954:53258568
290:204723181:52801394
289:224154852:83860557
328:225112127:79222615
314:229534754:72025568
341:230986394:74037001
301:232225841:76496609
323:232543812:65864275
349:233705389:90871911
291:233896316:54471226
297:248251191:86639661
306:252702129:75585270
326:256409415:92877256
354:257721983:55326428
312:258568377:52485047
342:258643305:73975429
299:260169148:84686721
352:261902489:69258113
287:261951031:74007587
300:262744353:56474883
351:264202897:76273788
343:274757715:80973409
310:274891710:65178811
320:277809328:80354835
286:279541571:75693030
338:283406269:51681107
313:287191020:83602719
322:287463260:58853608
302:290427711:92242878
319:297149758:52681662
305:298203338:83366451
311:299578801:54748152
295:300627372:82713868
350:311193844:61556361
344:312459246:62371640
330:320901282:76320380
327:322698654:85284994
```

318:325037484:67625177
345:335171470:58669804
325:341636154:79872257
353:345353683:58417567
298:359329976:56418127
333:365905095:51657663
303:378358072:67190457
340:385183731:77779221
308:385472266:52391853
296:386122900:83572648
360:389733016:74479887
304:391232990:84019873
337:391696645:56515939
324:394349528:80798564
335:394483964:67022166
336:420574208:59931272
357:421557660:60916866
359:427004918:79657806
339:435501383:67106624
361:439529478:59701601
356:468310243:78028894
331:474963218:71177924
329:480459887:80935472
355:480578743:89643207
332:481467342:63694395
307:518768356:60193840
309:522584924:75707080
358:536005054:63142397
334:539122998:77635900
315:574041699:59615109
284:583128307:66418919
316:608076476:55728783
283:618881588:92325735
348:619744548:54310638
292:620088958:58278653
294:631893091:55235901
346:640050035:56842020
285:671655292:78204690
317:737940990:57536962
293:833684640:80062866
347:853061185:64417121

Traitement lv all :

Pour des raisons de longueur du résultat, nous n'avons mis que les 50 premières lignes pour le résultat lv_all.csv

Pour ce traitement, il y a trois sorties : lv_all.csv, lv_all_minmax.csv et lv_all_minmax_graph.png. Nous ne passons par aucun ou plusieurs fichiers intermédiaires selon la sortie. Nous avons trié les tableaux suivants en fonction de ce schéma : "fichiers intermédiaires utiles pour le résultat suivant".

Première sortie : aucun fichier intermédiaire, tri direct :

- lv_all.csv

LV-Stations:Capacity:TotalConsumption(all)
163167:115326:312225
163037:115966:264778
163018:116110:276073
163156:116756:296384
163304:116998:321732
163226:117223:269161
163270:117543:313418
163371:117553:294914
163122:117661:257245
162966:118000:286094
162986:118027:244327
163076:118600:246418
162956:118766:264588
163043:118809:215662
163349:118816:292300
163115:119374:263893
163249:119545:300182
163130:119648:252503
163168:119812:317316
162970:120197:266729
163108:120231:324553
163084:120287:302337
163267:120336:226220
163171:120550:335100
162985:120670:286368
163109:120705:299832
163098:120724:244413
163285:120861:224586
163231:121157:326492
163054:121189:267444
163318:121228:302685
163200:121508:336947
162987:121721:236283
163057:122068:290557
163265:122275:226534
163068:122625:294367
162972:122768:264998
163066:122990:265104
163364:123298:308459
163028:123521:288264
163062:123530:325018
163063:123654:305074
163220:123951:253667
163354:123967:352670
163165:124204:301227
163091:124207:314882
163120:124498:246658
163040:124516:301847
163245:124624:232764

En plus de lv_all.csv, le traitement lv all a une deuxième sortie : lv_all_minmax.csv. Ce résultat rassemble les 10 postes avec le plus de consommation et les 10 avec le moins de consommation, triés en fonction de leur charge : du plus chargé au moins chargé. Pour arriver à cette sortie, nous passons par un fichier intermédiaire (temp_lv_all_minmax.csv) permettant de faire le tri en fonction de la quatrième colonne (qui correspond à la différence entre capacité et somme de la consommation d'un poste) et d'obtenir donc notre résultat final, après avoir supprimé la dernière colonne :

- temp_lv_all_minmax.csv	- lv_all_minmax.csv (résultat)
81115:1177774:191066:986708 171783:473189:193385:279804 173461:174295:194218:-19923 102937:760570:197342:563228 95122:867617:197645:669972 65819:287030:198203:88827 82057:639801:198695:441106 165545:606754:199106:407648 177008:416010:199135:216875 91030:754677:199165:555512 137529:924557:395328:529229 185803:401051:396334:4717 36825:312955:397898:-84943 161956:195368:399297:-203929 87528:842972:400803:442169 47752:243506:402128:-158622 175838:303839:404947:-101108 135364:717740:411076:306664 7114:729578:411362:318216 121215:1172729:413379:759350	LV-Stations:Capacity:TotalConsumption(all) 161956:195368:399297 47752:243506:402128 175838:303839:404947 36825:312955:397898 173461:174295:194218 185803:401051:396334 65819:287030:198203 177008:416010:199135 171783:473189:193385 135364:717740:411076 7114:729578:411362 165545:606754:199106 82057:639801:198695 87528:842972:400803 137529:924557:395328 91030:754677:199165 102937:760570:197342 95122:867617:197645 121215:1172729:413379 81115:1177774:191066

Enfin, la dernière sortie de ce traitement est un graphique représentant la consommation de 20 postes LV, mais *pas les mêmes que les 20 LV dans lv_min_max.csv !* Nous avons pris les 10 LV les plus chargés et les 10 LV les moins chargés (donc sans prendre en compte leur consommation). Ce graphique représente leurs consommations et leurs surconsommations par la couleur verte et rouge.

Pour y arriver, nous passons par trois fichiers intermédiaires nous permettant de récolter les informations nécessaires. *(Le résultat, donc le graphique, se trouve juste en dessous du tableau).*

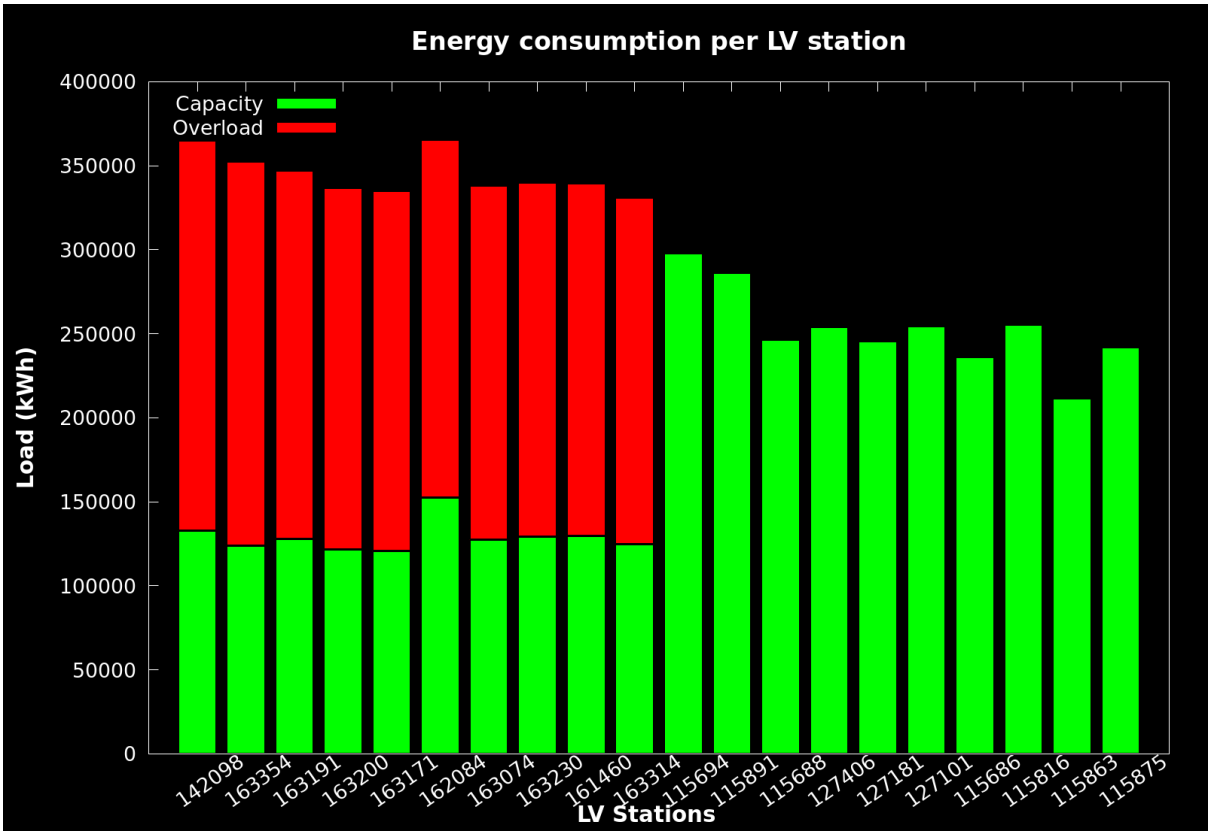
Le premier fichier intermédiaire (lv_info.csv) reprend lv_all.csv mais est trié en fonction de la différence entre la somme des consommations d'un poste et de sa capacité (cela est fait avec une colonne qui est ensuite supprimée) sans la ligne d'en-tête. *(Ici, seulement 50 lignes ont été mises en exemple car ce fichier est beaucoup trop long encore une fois).* Le deuxième fichier intermédiaire (lv_info_graph.csv) prend les 10 premiers postes et les 10 derniers du fichier

lv_info.csv, donc les 10 plus chargés et les 10 moins chargés. Enfin, nous avons le dernier fichier intermédiaire (lv_info_graph_with_parts.csv) qui nous permet de construire notre graphique. Il se constitue de deux colonnes construites grâce aux données du fichier intermédiaire n°2 : une colonne pour construire la partie de la consommation ‘verte’, et une colonne pour construire la partie de la surconsommation qui sera en rouge.

- lv_info.csv (temp 1)	- lv_info_graph.csv (temp 2)	- lv_info_graph_with_parts.csv (temp 3)
142098:132809:365143	142098:132809:365143	
163354:123967:352670	163354:123967:352670	132809:232334
163191:127927:347070	163191:127927:347070	123967:228703
163200:121508:336947	163200:121508:336947	127927:219143
163171:120550:335100	163171:120550:335100	121508:215439
162084:152418:365592	162084:152418:365592	120550:214550
163074:127253:338290	163074:127253:338290	152418:213174
163230:129081:340019	163230:129081:340019	127253:211037
161460:129629:339685	161460:129629:339685	129081:210938
163314:124800:330961	163314:124800:330961	129629:210056
161782:130563:336261	115694:3293605:297834	124800:206161
163231:121157:326492	115891:3285682:286262	297834:0
163304:116998:321732	115688:3248593:246422	286262:0
163108:120231:324553	127406:3262190:253879	246422:0
161956:195368:399297	127181:3257330:245482	253879:0
161788:129211:333099	127101:3268087:254312	245482:0
161851:146597:349887	115686:3258463:236031	254312:0
163373:134918:338017	115816:3278612:255581	236031:0
161491:129121:332197	115863:3236128:211367	255581:0
163062:123530:325018	115875:3270234:241924	211367:0
174448:142505:343690		241924:0
163015:133075:334225		
163137:133171:333390		
174659:144747:344743		
163812:155314:354848		
163168:119812:317316		
141771:137139:334624		
163167:115326:312225		
163009:131524:327838		
174526:143246:339285		
163270:117543:313418		
163809:156630:352193		
141963:136697:331335		
163328:134798:329236		
161639:142894:336603		
161544:126884:319978		
173904:171621:364563		
142203:130259:322594		
141803:164232:356552		
163181:134009:326261		
163247:156615:348505		
174684:143192:335056		
141971:143007:334244		
173707:174303:365433		
142185:145535:336437		
163192:143886:334777		

163091:124207:314882 163144:150345:340586 161710:143614:333716 163139:132447:32252		

Voici donc la sortie, après avoir envoyé nos données à gnuplot :



Comme dit auparavant, tous les autres traitements sont similaires aux deux traitements expliqués ci-dessus. Une variante de lv all serait lv all 2 par exemple, et nous passerions par le même nombre de fichiers intermédiaires pour chaque étape, et nous obtenons des résultats similaires (dans leur construction). Nous faisons simplement le tri par rapport à la centrale spécifiée.

De même, un traitement comme hvb comp, hvb comp 1, hva comp 5 ou lv comp sera tout comme le premier traitement expliqué : un tri direct et une seule sortie sera générée. Ce qui différera sera le tri qui sera par rapport à une centrale spécifiée ou non par l'utilisateur.

Merci d'avoir étudié notre projet.
groupA_MEF2
Wiame Boudella
Sanem Sayed
Sagana Srinivassane