Joël BOUGRON **15/11/22**
joel.bougron@ens.uvsq.fr

M1 CHPS
OBHPC
TD2
Programmation C et mesures de performances

A) PRESENTATION

Objet :

Il s'agit de réaliser un benchmark de la bande passante mémoire (et par extension de la latence mémoire).

Pour ce faire, nous répétons un relativement grand nombre de fois des opérations choisies dont on connait le nombre de cycles.

Nous faisons varier les paramètres indiqués ci-dessous.

Nous organisons les graphiques et les fichiers de performance de la manière suivante.

1) Histogrammes comparant les différentes versions pour chaque compilateur (confer annexe 1)

Procédure **dgemm_ijk** ::: 1 ensemble d'histogrammes

1 histogramme pour optimisation O0 // pas d'optimisation

1 barre pour compilateur gcc ::: **1 fichier de performance csv de 1 ligne**

1 barre pour compilateur clang

1 barre pour compilateur icx

1 histogramme pour optimisation O1 // 1er niveau

idem

1 histogramme pour optimisation O2 // 2ème niveau

idem

1 histogramme pour optimisation O3 // 3ème niveau

idem

1 histogramme pour optimisation Os // semblable à -O2 mais omtimisation de la taille du code plutôt que la vitesse

idem

1 histogramme pour optimisation Og // sauf ce qui interfère avec le débogage

idem

1 histogramme pour optimisation Ofast // semblage à O3 mais sans conformité exacte aux normes

idem

Procédure **dgemm_ikj** ::: idem

Procédure **dgemm_iex** ::: idem

Procédure **dgemm_unroll4** ::: idem

Procédure **dgemm_unroll8** ::: idem

Procédure **dgemm_cblas** ::: idem

Procédure **dotprod** ::: idem

Procédure **reduc** ::: idem

2) Histogrammes comparant les versions par compilateur (confer annexe 2)

Optimisation O0 ::: 1 ensemble d'histogrammes
- 1 histogramme pour compilateur gcc
  - 1 barre pour procédure **dgemm_ijk**
  - 1 barre pour procédure **dgemm_ikj**
  - 1 barre pour procédure **dgemm_iex**
  - 1 barre pour procédure **dgemm_unroll4**
  - 1 barre pour procédure **dgemm_unroll8**
  - 1 barre pour procédure **dgemm_cblas**
- 1 histogramme pour compilateur clang
  - idem
- 1 histogramme pour compilateur icc
  - idem
- 1 histogramme pour compilateur icx
  - idem

Optimisation O1 ::: idem

Optimisation O2 ::: idem

Optimisation O3 ::: idem

Optimisation Os ::: idem

Optimisation Og ::: idem

Optimisation Ofast ::: idem

Pour une mesure de performance stable :
- On a connecté le laptop est au secteur.
- On s'est assuré que le CPU tourne à une frequence stable en fixant la fréquence (ou le gouverneur) d'un ou plusieurs coeurs de calcul du CPU (**cpupower**).

$ cpupower frequency-info :
```
analyzing CPU 0:
  driver: intel_pstate
  CPUs which run at the same hardware frequency: 0
```

CPUs which need to have their frequency coordinated by software: 0
maximum transition latency: Cannot determine or is not supported.
hardware limits: 400 MHz - 4.60 GHz

available cpufreq governors: performance powersave

current policy: frequency should be within 400 MHz and 4.60 GHz.
La fréquence peut être choisie sur cet intervalle : nous voulons la valeur maximale : "4.60 HHz".
The governor "performance" may decide which speed to use
within this range.
Nous avons le choix entre deux governors : nous prenons "performance" pour maximiser la fréquence à "4.60 HHz".
current CPU frequency: Unable to call hardware
current CPU frequency: 4.37 GHz (asserted by call to kernel)
boost state support:
Supported: yes
Active: yes

$ sudo cpupower -c all frequency-set -g performance
sudo ::: because "Subcommand frequency-set needs root privileges"
frequency-set ::: cpupower's command
-g ::: new cpufreq governor (frequence-set's option)
performance ::: parmi les " available cpufreq governors"

- On pinne le processus sur le 4ème coeur de calcul.
- On arrête les autres processus.

Informations sur l'architecture cible (CPU) :

- $ lscpu

```
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  8
  On-line CPU(s) list:   0-7
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz
    CPU family:          6
    Model:               142
    Thread(s) per core:  2
    Core(s) per socket:  4
    Socket(s):           1
    Stepping:            12
    CPU max MHz:         4600,0000
    CPU min MHz:         400,0000
    BogoMIPS:            3999.93
    Flags:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush
                         dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_ts
                         c art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf
                          pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm p
                         cid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdra
                         nd lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single ssbd ibrs ibpb stibp
                          ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust
                         sgx bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt xsa
                         veopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_windo
                         w hwp_epp md_clear flush_l1d arch_capabilities
Virtualization features:
  Virtualization:        VT-x
Caches (sum of all):
  L1d:                   128 KiB (4 instances)
  L1i:                   128 KiB (4 instances)
  L2:                    1 MiB (4 instances)
  L3:                    8 MiB (1 instance)
NUMA:
  NUMA node(s):          1
  NUMA node0 CPU(s):     0-7
Vulnerabilities:
  Itlb multihit:         KVM: Mitigation: VMX disabled
  L1tf:                  Not affected
  Mds:                   Not affected
  Meltdown:              Not affected
  Mmio stale data:       Mitigation; Clear CPU buffers; SMT vulnerable
  Spec store bypass:     Mitigation; Speculative Store Bypass disabled via prctl and seccomp
  Spectre v1:            Mitigation; usercopy/swapgs barriers and __user pointer sanitization
  Spectre v2:            Mitigation; Enhanced IBRS, IBPB conditional, RSB filling
  Srbds:                 Mitigation; Microcode
  Tsx async abort:       Not affected
```

- $ cat /proc/cpuinfo

```
processor : 4
vendor_id : GenuineIntel
cpu family: 6
model                   : 142
model name              : Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz
stepping   : 12
microcode: 0xf0
cpu MHz                 : 2000.000
cache size: 8192 KB
physical id             : 0
siblings    : 8
core id                 : 0
cpu cores : 4
```

apicid : 1
initial apicid : 1
fpu : yes
fpu_exception : yes
cpuid level : 22
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb invpcid_single ssbd ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust sgx bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt xsavec xgetbv1 xsaves dtherm ida arat pln pts hwp hwp_notify hwp_act_window hwp_epp md_clear flush_l1d arch_capabilities
vmx flags : vnmi preemption_timer invvpid ept_x_only ept_ad ept_1gb flexpriority tsc_offset vtpr mtf vapic ept vpid unrestricted_guest ple pml ept_mode_based_exec
bugs : spectre_v1 spectre_v2 spec_store_bypass swapgs itlb_multihit srbds mmio_stale_data
bogomips : 3999.93
clflush size : 64
cache_alignment : 64
address sizes : 39 bits physical, 48 bits virtual
power management:

Informations sur l'architecture cible (caches de données) :

- Dans le chemin **/sys/devices/system/cpu/cpu0/cache/index0/*** pour le cache L1 :

coherency_line_size : 64
id : 0
level : 1
number_of_sets : 64
physical_line_partition : 1
shared_cpu_list : 0,4
shared_cpu_map : 11
size : 32K
type : Data
ways_of_associativity : 8

- Dans le chemin **/sys/devices/system/cpu/cpu0/cache/index2/*** pour le cache L2 :

coherency_line_size : 64
id : 0
level : 2
number_of_sets : 1024
physical_line_partition : 1
shared_cpu_list : 0,4
shared_cpu_map : 11
size : 256K
type : Unified
ways_of_associativity : 4

- Dans le chemin **/sys/devices/system/cpu/cpu0/cache/index3/*** pour le cache L3 :

coherency_line_size : 64
id : 0
level : 3
number_of_sets : 8192
physical_line_partition : 1
shared_cpu_list : 0-7
shared_cpu_map : ff
size : 8192K
type : Unified
ways_of_associativity : 16

Autres informations (RAM) :

- Dans le chemin **/proc/meminfo** (ou commande $ free) :

MemTotal: 16026644 kB
iMemFree: 2410088 kB

B) ANALYSE DES GRAPHIQUES

Il s'agit de comparer des bandes passantes mémoire (en MiB/s). Une meilleure performance correspond à une bande passante élevée et une faible latence mémoire (durée d'accès à une adresse mémoire).

En effet :

- **La fréquence (GHz) est rendue constante.**
- **Le nombre de cycles est connu, selon la taille des matrices (fonctions de la dimension n et des tailles des flottants) et le nombre de kernel repetitions (r).**
- **Le nombre de cycles d'horloge est mesuré. La durée d'un cycle d'horloge est connue.**
- **Les performances sont réduites par la complexité algorithmique. D'où de meilleures performances avec l'invariant extraction (ikj) et avec le loop unrolling (dgemm unroll8).**

- **Presque toutes les versions de calcul sont plus performantes avec une optimisation (de O1 à Ofast) que sans optimisation. En effet, c'est dans la définition d'une optimisation de compilation.**

- **Exception : dgemm cblas ne semble pas être améliorée pas une optimisation. En effet, le principal objet de la bibliothèque CBLAS est de permettre intrinsèquement un calcul performant.**

- **Pour la version de calcul naïve (dgemm_ijk), GCC présente une moindre performance pour Os et Og. Donc GCC semble privilégier la petite taille de code (Os) et le débogage (Og) par rapport à l'optimisation.**

- **CBLAS permet une bande passante mémoire environ 4,5 fois plus élevée que chacune des autres versions.**

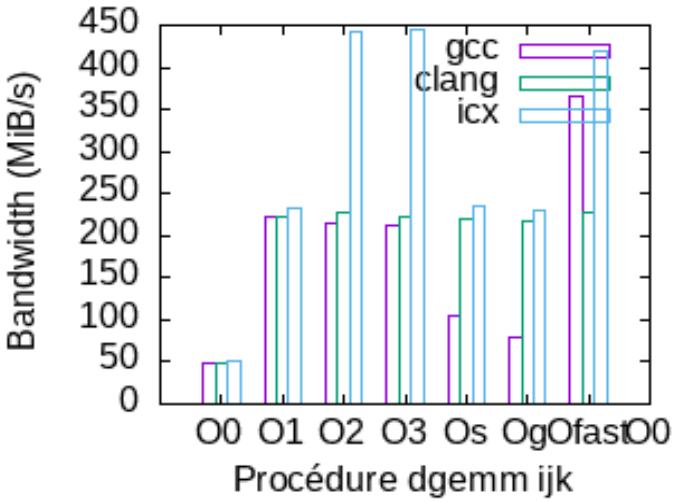
- **C'est avec ICX que CBLAS est de loin la plus avantageuse.**

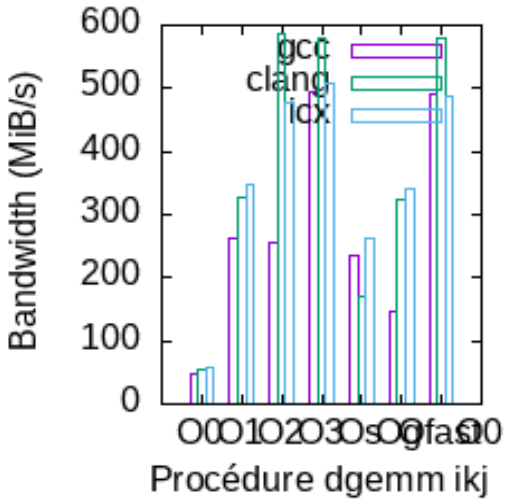
- **Pour les versions de calcul les plus naïves (dgemm_ijk, et dans une moindre mesure dotprod et reduc), pour O2 et P3, ICX permet un saut avantageux de performance.**

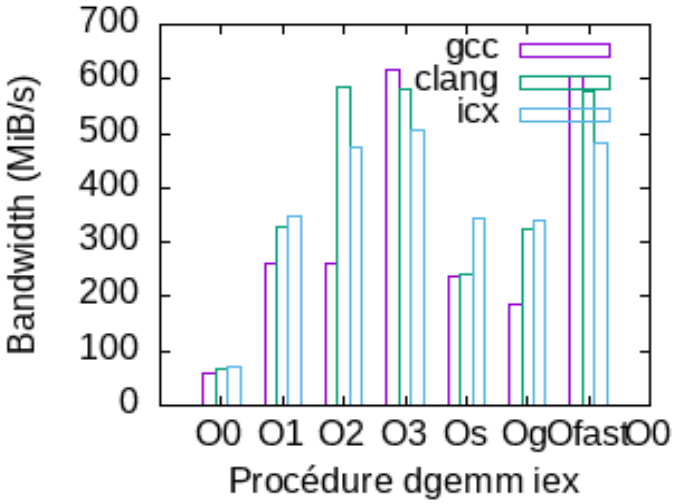
- **ICX apporte moins d'avantages avec Ofast qu'avec les autres flags d'optimisation.**

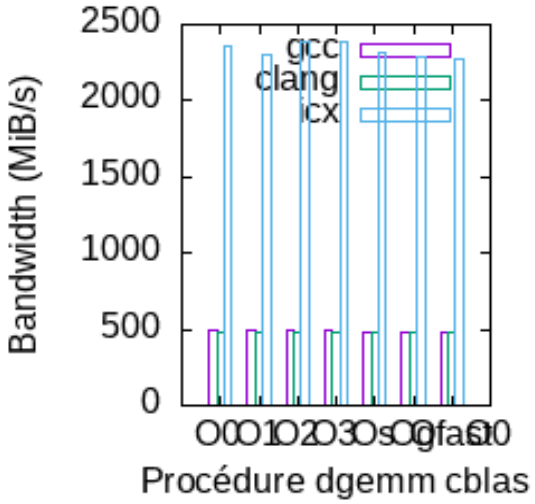
ANNEXE 1

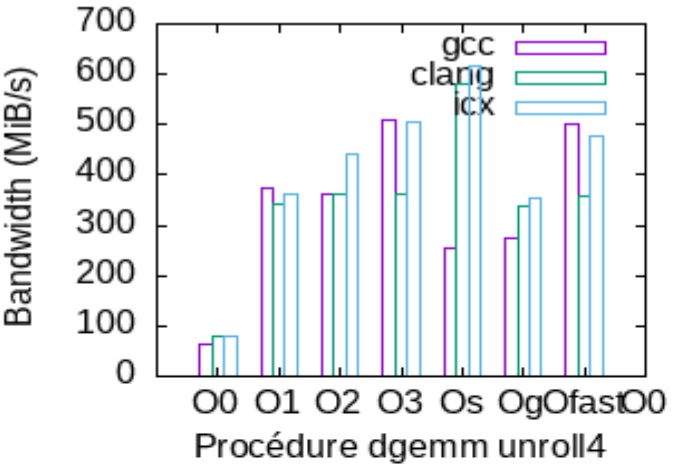
Histogrammes comparant les différentes versions pour chaque compilateur

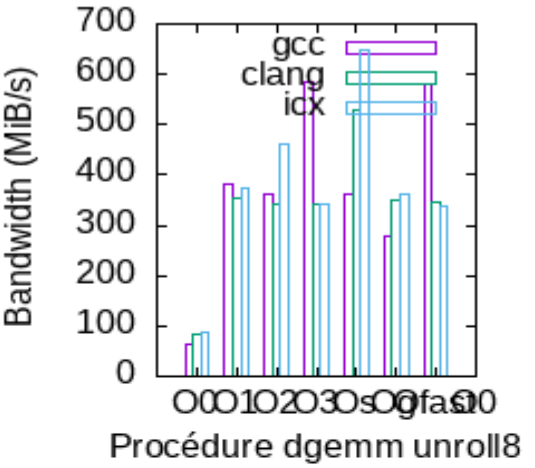

Bandwidth (MiB/s)
gcc
clang
icx
O0 O1 O2 O3 Os Og Ofast O0
Procédure dgemm ijk

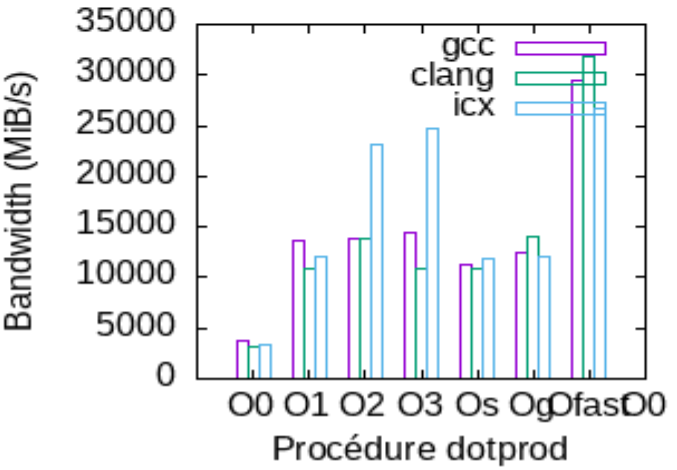

Bandwidth (MiB/s)
gcc
clang
icx
O0 O1 O2 O3 Os Og Ofast O0
Procédure dgemm ikj

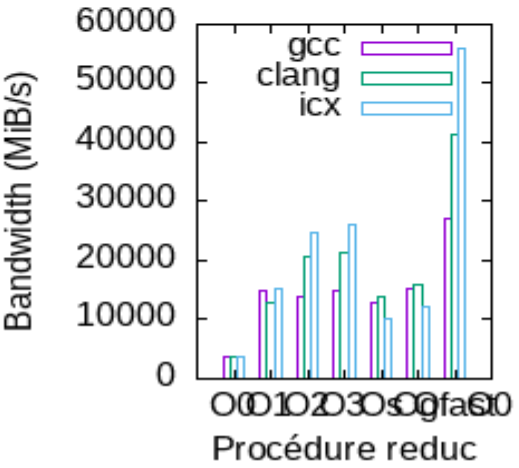

Bandwidth (MiB/s)
gcc
clang
icx
O0 O1 O2 O3 Os Og Ofast O0
Procédure dgemm iex


Bandwidth (MiB/s)
gcc
clang
icx
O0 O1 O2 O3 Os Og Ofast O0
Procédure dgemm cblas


Bandwidth (MiB/s)
gcc
clang
icx
O0 O1 O2 O3 Os Og Ofast O0
Procédure dgemm unroll4

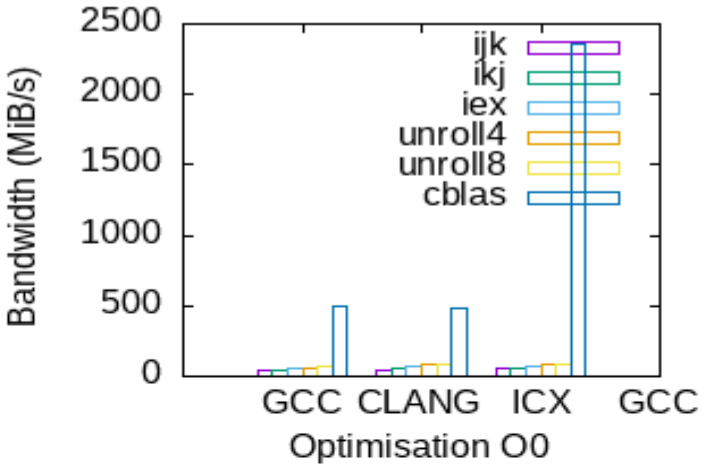

Bandwidth (MiB/s)
gcc
clang
icx
O0 O1 O2 O3 Os Og Ofast O0
Procédure dgemm unroll8

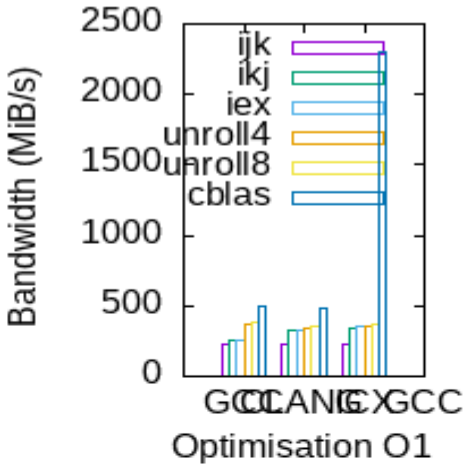

Bandwidth (MiB/s)
gcc
clang
icx
O0 O1 O2 O3 Os Og Ofast O0
Procédure dotprod

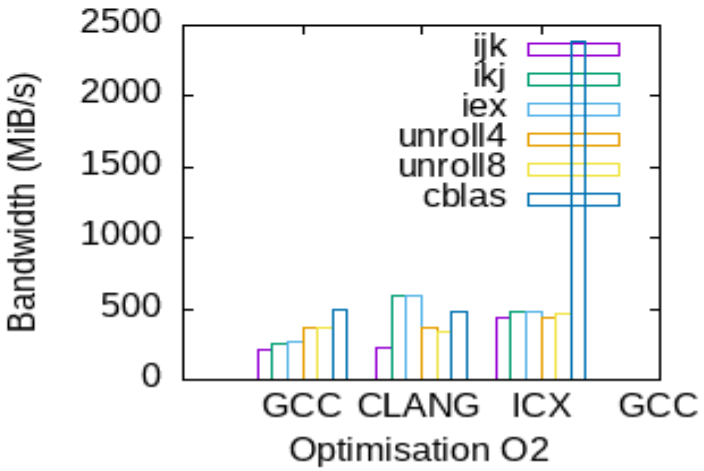

Bandwidth (MiB/s)
gcc
clang
icx
O0 O1 O2 O3 Os Og Ofast O0
Procédure reduc

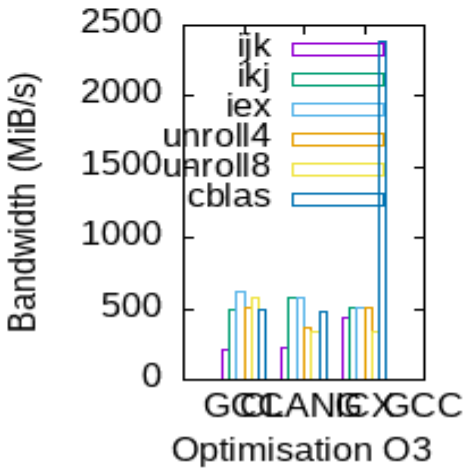
ANNEXE 2

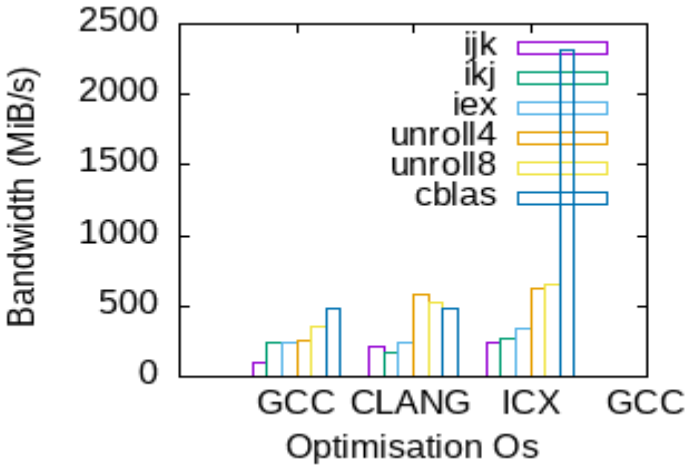
Histogrammes comparant les comparant les versions par compilateur

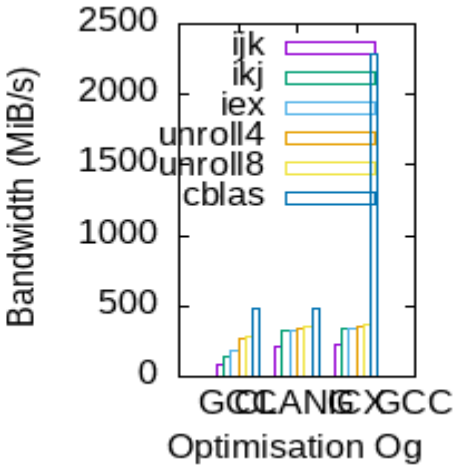

Bandwidth (MiB/s)
2500
2000
1500
1000
500
0
ijk
ikj
iex
unroll4
unroll8
cblas
GCC
CLANG
ICX
GCC
Optimisation O0

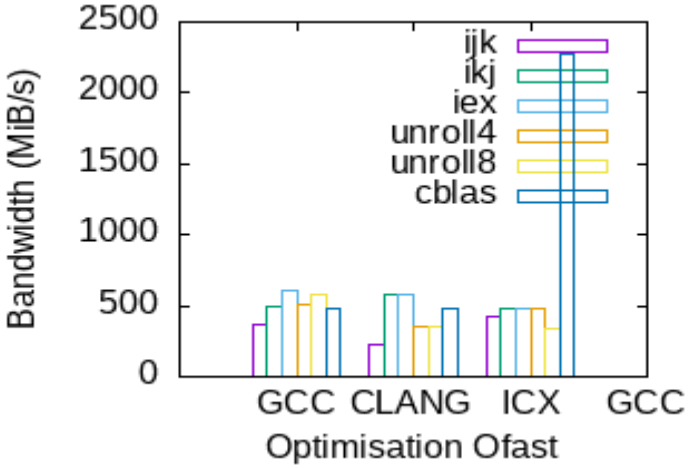

Bandwidth (MiB/s)
2500
2000
1500
1000
500
0
ijk
ikj
iex
unroll4
unroll8
cblas
GCC
CLANG
ICX
GCC
Optimisation O1


Bandwidth (MiB/s)
2500
2000
1500
1000
500
0
ijk
ikj
iex
unroll4
unroll8
cblas
GCC
CLANG
ICX
GCC
Optimisation O2


Bandwidth (MiB/s)
2500
2000
1500
1000
500
0
ijk
ikj
iex
unroll4
unroll8
cblas
GCC
CLANG
ICX
GCC
Optimisation O3


Bandwidth (MiB/s)
2500
2000
1500
1000
500
0
ijk
ikj
iex
unroll4
unroll8
cblas
GCC
CLANG
ICX
GCC
Optimisation Os


Bandwidth (MiB/s)
2500
2000
1500
1000
500
0
ijk
ikj
iex
unroll4
unroll8
cblas
GCC
CLANG
ICX
GCC
Optimisation Og


Bandwidth (MiB/s)
2500
2000
1500
1000
500
0
ijk
ikj
iex
unroll4
unroll8
cblas
GCC
CLANG
ICX
GCC
Optimisation Ofast