Implementações dos *Benchmarks* EP e IS em Rust

Ricardo Padilha

Resumindo o *benchmark* EP

- Gera um array de pares de valores aleatórios
  a. *Randlc* : gerador de números aleatórios
  b. *Vranlc*: usa *Randlc* para preencher vetor com números aleatórios
  c. *Verify*: método utilizado para verificar o quão corretos os cálculos de valores esperados estão
- Tempos medidos:
  a. Tempo total de execução
  b. Tempo para gerar valores aleatórios
  c. Tempo para verificar aceitabilidade dos cálculos

Principais dificuldades no EP

- Familiaridade com Fortran
- Tradução literal não foi possível
    a. Optei por utilizar um canal de comunicação entre threads como método similar a "*parallel reduction*" disponível em fortran
- Ajuste no tamanho da stack do programa para poder computar vetores maiores
- Como foi a primeira implementação, os conceitos de "*thread-safety*" de Rust ainda não eram totalmente compreendidos

Exemplo de Implementação: seção paralela

```
for tid: usize in 0..num_threads {
    let nk: usize = (NK / num_threads) +1;//tamanho dos 'chunks'
    let start: usize = tid * nk;//offset do chunk de cada thread
    let end: usize = (tid+1) * nk;//offset do chunk de cada thread
    let tx2_clone: Sender<f64> = tx2.clone();
    let tx1_clone: Sender<f64> = tx1.clone();

    threads.push(thread::spawn(move || {
        let mut sx: f64 = 0.0;
        let mut sy: f64 = 0.0;

        let s: f64 = 31269.0;
        let an: f64 = 132608.0;
        let mut t1: f64;
        let mut t2: f64;
        let mut t3: f64;
        let mut t4: f64;
        let mut ik: usize;
        let mut x1: f64;
        let mut x2: f64;
        let mut l: usize;
        let mut qq: [f64; NQ] = [0.0; NQ];

        for k: usize in start..end {
            let mut kk: usize = start + k;
            t1 = s;
            t2 = an;

            // Find starting seed t1 for this kk.
            //timer 2 starts here
            let start_time2: Instant = Instant::now();
            for _i: i32 in 1..=100 {
                ik = kk / 2;
                if 2 * ik != kk {
                    t3 = auxfunctions::randlc(x: &mut t1, a: t2);
                }
                if ik == 0 {
                    break;
                }
                let t2copy: f64 = t2;
                t3 = auxfunctions::randlc(x: &mut t2, a: t2copy);
                kk = ik;
            }
```

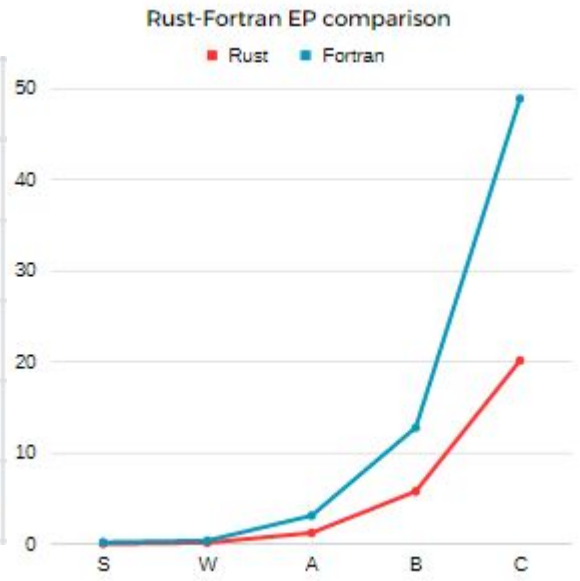
```
            // Compute uniform pseudorandom numbers.
            auxfunctions::vranlc(n: 2 * nk as i32, x: t1, a: an,  y: &mut x);

            //timer 2 ends here
            let elapsed_time2: f64 = start_time2.elapsed().as_secs_f64();
            tx2_clone.send(elapsed_time2).unwrap();

            //timer 1 starts here
            let start_time1: Instant = Instant::now();
            // Compute Gaussian deviates by acceptance-rejection method and
            // tally counts in concentric square annuli.
            for i: usize in 0..nk {
                x1 = 2.0 * x[2 * i] - 1.0;
                x2 = 2.0 * x[2 * i + 1] - 1.0;
                t1 = x1.powi(2) + x2.powi(2);

                if t1 <= 1.0 {
                    t2 = (-2.0 * t1.ln() / t1).sqrt();
                    t3 = x1 * t2;
                    t4 = x2 * t2;
                    l = t3.abs().max(t4.abs()) as usize;
                    qq[l] += 1.0;
                    sx += t3;
                    sy += t4;
                }
            }

            //timer 1 ends here
            let elapsed_time1: f64 = start_time1.elapsed().as_secs_f64();
            tx1_clone.send(elapsed_time1).unwrap();
        }

        (sx, sy, qq)
    }));
}
for thread: JoinHandle<(f64, f64, [f64; 10])> in threads {
    let (sx_thread: f64, sy_thread: f64, qq_thread: [f64; 10]) = thread.join().unwrap();
    sx += sx_thread;
    sy += sy_thread;
    for i: usize in 0..NQ {
        q[i] += qq_thread[i];
    }
}
```

Resultados do EP

Média de 10 execuções no equipamento

- I5-4210U (2014)
- 2 cores de 1.7 GHz
- 8 GB de RAM

	Rust	Fortran
S	0.01	0.2
W	0.15	0.38
A	1.23	3.12
B	5.79	12.8
C	20.15	48.9



Resumindo o *benchmark* IS

- Gera um array de inteiros e os ordena usando um algoritmo de ordenação paralelo, seja ele *bucket-sort* ou *radix-sort*
  a. *Randlc* : gerador de números aleatórios
  b. *Find_my_seed* : usa *randlc* para gerar uma *seed* diferente para cada thread
  c. *Create_seq*: preenche *array* com inteiros aleatórios usando as funções "*find_my_seed*" e "*randlc*"
  d. *Alloc_key_buff*: Na implementação oficial, aloca a memória necessária para os *buckets* usados para ordenação
  e. *Rank*: chamada uma vez para inicializar os dados, páginas e tabelas, depois é chamada uma vez para cada iteração, definida nas macros. Separa os inteiros em "*buckets*" apropriados.
  f. *Full_verify*: ordena os "*buckets*" (e, por consequência, o *array*) e verifica se foram ordenados corretamente
- Tempos medidos:
  a. Tempo total de execução
  b. Tempo para gerar valores aleatórios
  c. Tempo para separar valores nos "*buckets*" apropriados
  d. Tempo para ordenar e verificar a solução

Principais Dificuldades

- Diretiva *omp* não existe em Rust
  - Encontrada alternativa com *std::thread*
- Implementação em C utiliza ponteiros
  - Alternativa com vetores dinâmicos e referências mutáveis
- Substituição das macros de C
  - Alternativa através da passagem de parâmetros para cada função
- Permissões de alocação de memória
  - Explicado mais adiante

Exemplo de Implementação: *parallel_bucket_sort*

```rust
fn parallel_bucket_sort(arr: Vec<i64>, num_buckets: usize) -> Vec<i64> {
    let len: usize = arr.len();

    // create an array of empty buckets
    let mut buckets: Vec<Vec<i64>> = vec![vec![]; num_buckets];

    //this is what the "rank" function does in the C program, separating the values into the appropriate buckets
    // scatter the values from the input array into the buckets
    for i: usize in 0..len {
        let bucket_idx: usize = (arr[i] as f64 * num_buckets as f64 / std::i64::MAX as f64) as usize;
        buckets[bucket_idx].push(arr[i]);
    }

    // sort each bucket using multiple threads
    //this is what the "full_verify" function does in the C program, it sorts the keys, then verifies that they are in order
    let mut handles: Vec<JoinHandle<Vec<i64>>> = Vec::new();
    for i: usize in 0..num_buckets {
        let bucket: Vec<i64> = std::mem::take(dest: &mut buckets[i]);
        let handle: JoinHandle<Vec<i64>> = thread::spawn(move || {
            let mut bucket: Vec<i64> = bucket;
            bucket.sort();
            bucket
        });
        handles.push(handle);
    }

    // concatenate the sorted buckets back into a single array
    let mut sorted_vec: Vec<i64> = Vec::with_capacity(len);
    for handle: JoinHandle<Vec<i64>> in handles {
        let bucket: Vec<i64> = handle.join().unwrap();
        sorted_vec.extend(iter: bucket);
    }

    sorted_vec
} fn parallel_bucket_sort
```

```rust
//function used to verify the if the sorting was correct
fn verify_sort(vec: &Vec<i64>) -> i64 {
    let mut count: i64 = 0;

    for i: usize in 1..vec.len() {
        if vec[i - 1] > vec[i] {
            count += 1;
        }
    }

    count
}
```

Problemas de “*make*” no NPB oficial classe D

```
cd IS; make CLASS=D
make[1]: Entering directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/IS'
make[2]: Entering directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/sys'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/sys'
../sys/setparams is D
gcc -c  -O3 -fopenmp is.c
gcc -O3 -fopenmp -o ../bin/is.D.x is.o ../common/c_print_results.o ../common/c_timers.o ../common/c_wtime.o -lm
is.o: in function `alloc_key_buff._omp_fn.0':
is.c:(.text+0x3f): relocation truncated to fit: R_X86_64_PC32 against symbol `key_buff2' defined in COMMON section in is.o
is.o: in function `full_verify._omp_fn.0':
is.c:(.text+0x149): relocation truncated to fit: R_X86_64_PC32 against symbol `key_buff2' defined in COMMON section in is.o
is.c:(.text+0x16d): relocation truncated to fit: R_X86_64_PC32 against symbol `key_buff_ptr_global' defined in COMMON section in is.o
is.o: in function `rank._omp_fn.0':
is.c:(.text+0x464): relocation truncated to fit: R_X86_64_PC32 against symbol `key_buff2' defined in COMMON section in is.o
is.o: in function `full_verify':
is.c:(.text+0xc86): relocation truncated to fit: R_X86_64_PC32 against symbol `passed_verification' defined in COMMON section in is.o
is.o: in function `rank':
is.c:(.text+0xd21): relocation truncated to fit: R_X86_64_PC32 against symbol `test_index_array' defined in COMMON section in is.o
is.c:(.text+0xd3d): relocation truncated to fit: R_X86_64_PC32 against symbol `test_index_array' defined in COMMON section in is.o
is.c:(.text+0xd4f): relocation truncated to fit: R_X86_64_PC32 against symbol `test_index_array' defined in COMMON section in is.o
is.c:(.text+0xd61): relocation truncated to fit: R_X86_64_PC32 against symbol `test_index_array' defined in COMMON section in is.o
is.c:(.text+0xd73): relocation truncated to fit: R_X86_64_PC32 against symbol `test_index_array' defined in COMMON section in is.o
is.c:(.text+0xd87): additional relocation overflows omitted from the output
collect2: error: ld returned 1 exit status
make[1]: *** [Makefile:16: ../bin/is.D.x] Error 1
make[1]: Leaving directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/IS'
make: *** [Makefile:31: is] Error 2
```

Problemas de “*make*” no NPB oficial classe E

```
cd IS; make CLASS=E
make[1]: Entering directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/IS'
make[2]: Entering directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/sys'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/sys'
../sys/setparams is E
gcc -c  -O3 -fopenmp is.c
gcc -O3 -fopenmp -o ../bin/is.E.x is.o ../common/c_print_results.o ../common/c_timers.o ../common/c_wtime.o -lm
is.o: in function `alloc_key_buff._omp_fn.0':
is.c:(.text+0x44): relocation truncated to fit: R_X86_64_PC32 against symbol `key_buff2' defined in COMMON section in is.o
is.o: in function `full_verify._omp_fn.1':
is.c:(.text+0xb6): relocation truncated to fit: R_X86_64_PC32 against symbol `key_array' defined in COMMON section in is.o
is.o: in function `full_verify._omp_fn.0':
is.c:(.text+0x159): relocation truncated to fit: R_X86_64_PC32 against symbol `key_buff2' defined in COMMON section in is.o
is.c:(.text+0x160): relocation truncated to fit: R_X86_64_PC32 against symbol `key_array' defined in COMMON section in is.o
is.c:(.text+0x17d): relocation truncated to fit: R_X86_64_PC32 against symbol `key_buff_ptr_global' defined in COMMON section in is.o
is.o: in function `rank._omp_fn.0':
is.c:(.text+0x33e): relocation truncated to fit: R_X86_64_PC32 against symbol `key_array' defined in COMMON section in is.o
is.c:(.text+0x468): relocation truncated to fit: R_X86_64_PC32 against symbol `key_array' defined in COMMON section in is.o
is.c:(.text+0x474): relocation truncated to fit: R_X86_64_PC32 against symbol `key_buff2' defined in COMMON section in is.o
is.o: in function `create_seq._omp_fn.0':
is.c:(.text+0x9b3): relocation truncated to fit: R_X86_64_PC32 against symbol `key_array' defined in COMMON section in is.o
is.o: in function `full_verify':
is.c:(.text+0xca6): relocation truncated to fit: R_X86_64_PC32 against symbol `passed_verification' defined in COMMON section in is.o
is.o: in function `rank':
is.c:(.text+0xd25): additional relocation overflows omitted from the output
collect2: error: ld returned 1 exit status
make[1]: *** [Makefile:16: ../bin/is.E.x] Error 1
make[1]: Leaving directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/IS'
make: *** [Makefile:31: is] Error 2
```

Problemas de “*make*” no NPB oficial classe D e E

Consertados ao adicionar “*flag*” de compilação “-fPIC” ao arquivo “*make.def*”

- A opção fPIC no GCC permite que o endereço das bibliotecas compartilhadas seja relativo para que o executável seja independente da posição das bibliotecas. Isso permite compartilhar bibliotecas construídas que possuem dependências de outras bibliotecas compartilhadas.
- fPIC significa "force Position Independent Code"

Resultado: Erros Diferentes

```
cd IS; make CLASS=D
make[1]: Entering directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/IS'
make[2]: Entering directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/sys'
make[2]: Nothing to be done for 'all'.
make[2]: Leaving directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/sys'
../sys/setparams is D
gcc -c  -O3 -fopenmp -fPIC is.c
gcc -O3 -fopenmp -fPIC -o ../bin/is.D.x is.o ../common/c_print_results.o ../common/c_timers.o ../common/c_wtime.o -lm
/usr/bin/ld: failed to convert GOTPCREL relocation; relink with --no-relax
collect2: error: ld returned 1 exit status
make[1]: *** [Makefile:16: ../bin/is.D.x] Error 1
make[1]: Leaving directory '/home/rfpadilha/NPB3.4.2/NPB3.4-OMP/IS'
make: *** [Makefile:31: is] Error 2
```

Problemas de “*make*” no NPB oficial classe D e E

Consertados ao adicionar “*flag*” de compilação “-mcmodel=large” ao arquivo “*make.def*”

- Diz ao compilador para usar um modelo de memória específico para gerar código e armazenar dados.
- “*Large*”: Não coloca nenhuma restrição de memória em código ou dados. Todos os acessos de código e dados devem ser feitos com endereçamento absoluto.

Compilado com sucesso após a adição.

Problemas de execução nas classes D e E em C

```
rfpadilha@DESKTOP-1GSPPCC:~/NPB3.4.2/NPB3.4-OMP/bin$ ./is.D.x
-bash: ./is.D.x: Cannot allocate memory
rfpadilha@DESKTOP-1GSPPCC:~/NPB3.4.2/NPB3.4-OMP/bin$ ./is.E.x
-bash: ./is.E.x: Cannot allocate memory
rfpadilha@DESKTOP-1GSPPCC:~/NPB3.4.2/NPB3.4-OMP/bin$
```

Problemas de Execução na classe E em Rust

```
memory allocation of 274877906944 bytes failed
error: process didn't exit successfully: `target\debug\IS.exe E` (exit code: 0xc0000409, STATUS_STACK_BUFFER_OVERRUN)
```

Faz sentido ser impossível alocar 274 GB de memória, dado que a máquina não possui essa capacidade.

Resultados

Todas as ordenações foram feitas com sucesso, sem elementos fora de ordem.
Média de 10 execuções no equipamento

- I5-4210U (2014)
- 2 cores de 1.7 GHz
- 8 GB de RAM

	Rust	C
S	0.9	0.01
W	1.89	0.12
A	1.97	0.5
B	2.14	1.99
C	1.85	8.71
D	2.01